Juez para el aprendizaje de bases de datos

A judge for database learning

Por

Iker Burgoa Muñoz

Tamara Huertas Smolis

Daniel Ibáñez Padial

Iván Ruiz Quintana

UNIVERSIDAD COMPLUTENSE
MADRID

Grado en Ingeniería de Computadores

Grado en Ingeniería Informática

Grado en Ingeniería del Software

FACULTAD DE INFORMÁTICA

Directores

Enrique Martín Martín

Jesús Correas Fernández

Madrid, 2020-2021

Dedicatorias

A mi familia por el apoyo dado siempre aguantando mi ritmo de no parar por casa y a todos mis amigos por apoyarme en todo momento y hacer que este camino hasta aquí haya sido mucho más llevadero. Gracias a todos.

- *Iker Burgoa Muñoz*

A mi familia y en particular a mis hermanas por todo el apoyo, el ánimo que me han dado y por la confianza que han depositado en mí. También, a mis amigos por estar en todo momento a mi lado. Muchas gracias.

- *Tamara Huertas Smolis*

A mi novia por ser apoyo moral y por darme ese ánimo necesario para sacar esto adelante. A mi familia por aguantar mi mal humor en los momentos de estrés. Gracias.

- *Daniel Ibáñez Padial*

A mi familia por confiar en mí incluso en los malos momentos y a mis amigos por haber estado ahí siempre que lo he necesitado. Os lo agradezco.

- *Iván Ruiz Quintana*

Agradecimientos

Muchas gracias a todas las personas que nos han acompañado durante el desarrollo de este proyecto. Gracias a los tutores por guiarnos y ayudarnos en todo el proceso.

Resumen

El uso de los jueces automáticos en la Facultad de Informática de la Universidad Complutense de Madrid (UCM) ha logrado un importante avance a la hora de ayudar con la enseñanza de la programación. Sin embargo, dichas herramientas no han conseguido un impacto equivalente en otros centros, por lo que, a día de hoy, la gran mayoría de facultades y universidades continúan impartiendo la programación en papel, con la consecuente lentitud que ello puede acarrear en la retroalimentación entre profesor y alumno.

Learn SQL es un corrector automático para el aprendizaje de lenguaje SQL, desarrollado con el *framework* de Django y sobre Oracle, que surge, entre otras razones, para fomentar la participación del alumnado. Este trabajo, en base al uso de tales jueces para el aprendizaje de base de datos, busca ver cómo pueden desarrollarse tales herramientas y analizar sus principales características (capacidad multidioma, sistema de pistas, logros y *feedback* al enviarse una solución); todo ello tiene un fin último, la posibilidad de ser extendida con nuevas funcionalidades en el futuro.

Palabras clave

Open Source, bases de datos, enseñanza de la programación, jueces de programación, ludificación, sistemas automáticos de corrección de ejercicios

Abstract

The use of automatic judges in the Faculty of Computer Science at the Complutense University of Madrid (UCM) has made significant progress in helping with the teaching of programming. However, these tools have not had an equivalent impact in other centres, so that, to this day, the vast majority of faculties and universities continue to teach on paper, with the consequent slowness that this can cause in the feedback between teacher and student.

Learn SQL is an automatic checker for learning SQL language, developed with the Django framework and on Oracle, which arises, among other reasons, to encourage student participation. This work, based on the use of such judges for database learning, seeks to see how such tools can be developed and to analyse their main features (multi-language capability, hint system, achievements and feedback when a solution is sent); all of this has an ultimate goal, the possibility of being extended with new functionalities in the future.

Keywords

Open Source, databases, programming teaching, programming judges, gamification, automatic exercise correction systems.

Índice general

Resumen **5**

Abstract **6**

1. Introducción **9**
1.1 Motivación 9
1.2 Objetivos 10
1.3 Plan de Trabajo 12

2. Introduction **14**
2.1 Motivation 14
2.2 Objectives 15
2.3 Work Plan 17

3. Herramientas utilizadas en el proyecto **19**
3.1 Django 19
3.2 Oracle y PostgreSQL 19
3.3 Pylint y Codecov 20
3.4 Editores de texto 21
3.5 Herramientas de comunicación y gestión de proyecto 21
3.6 Diseño de bocetos 23

4. Plataformas antecesoras **24**
4.1 Jueces y sistemas automáticos de evaluación 24
4.2 Herramientas para el aprendizaje de SQL 27

5. Estado inicial de Learn SQL **29**

6. Metodología de desarrollo **33**

7. Implementaciones **36**
7.1 Botón para descargar script de tablas 36
7.2 Mejora de la respuesta generada ante una solución incorrecta en el esquema generado 38
7.3 Grupos de clase 40
7.4 Vista Resultados 43
7.5 Enlace tabla clasificación envíos 49
7.6 Colapsar Tablas 51
7.7 Descargar Envíos 52
7.8 Campo para enviar un archivo como respuesta 54
7.9 Filtro de fechas en la vista de resultados 56
7.10 Podio 58
7.11 Mostrar tablas modificadas 59

7.12 Descargar ranking 61
7.13 Logros 63
7.14 Bases de datos iniciales 72
7.15 Ejercicios de discriminación de consultas 76
7.16 Soporte multidioma 78
7.17 Pistas 81
7.18 Informar al usuario de los logros conseguidos 91

8. Contribuciones 93
8.1 Iker Burgoa Muñoz 93
8.2 Tamara Huertas Smolis 94
8.3 Daniel Ibáñez Padial 97
8.4 Iván Ruiz Quintana 99

9. Conclusiones y trabajo futuro 101

10. Conclusions and future work 104

Bibliografía 107

1. Introducción

1.1 Motivación

Actualmente, en las universidades de toda España existen asignaturas relacionadas con lenguajes de programación cuyos trabajos, ejercicios y otras partes prácticas siguen siendo realizadas en papel. Esto hace que la corrección por parte de los profesores lleve más tiempo y, por consiguiente, la respuesta recibida por los alumnos, el *feedback*, tarde más.

Así, para tratar de agilizar los plazos, desde hace unos años, en la Facultad de Informática de la Universidad Complutense de Madrid (UCM) se usa un juez de programación para varias asignaturas de programación y algoritmia, como en el caso de Fundamentos de la Programación, Fundamentos de la Algoritmia y Estructura de Datos. Sin embargo, aunque con este tipo de sistemas el alumno recibe retroalimentación inmediata de la respuesta dada a los problemas propuestos, el resultado obtenido por el sistema suele ser bastante genérico y no aporta demasiada información.

Un aspecto importante de este tipo de aplicaciones de corrección automática de ejercicios es la posibilidad de obtener resultados estadísticos sobre la actividad de los alumnos que se pueden aprovechar para mejorar la docencia en los siguientes años. Por ejemplo, se puede obtener el número total de ejercicios enviados por los alumnos, los periodos en los que más envíos se han hecho, los problemas que han requerido un mayor número de intentos, en qué parte del curso hay un mayor número de alumnos que deciden abandonar la asignatura, etc. Esta información puede ser muy valiosa para reforzar aquellos elementos del contenido del curso que suponen un mayor esfuerzo para los alumnos; en algunas aplicaciones estas estadísticas son visibles para el alumnado, que puede ver en qué puesto están tanto ellos como el resto de compañeros. Con este sistema se busca motivarlos al ver que consiguen “medallas” como recompensa por haber logrado resolver un número

determinado de ejercicios. Esto hace que esa “recompensa” sea un incentivo para seguir con la asignatura e ir resolviendo más ejercicios.

Los jueces de programación se han usado en algunas asignaturas de la facultad para la realización de exámenes o pruebas evaluatorias. Por parte del alumno se está programando en una herramienta conocida; para el profesor los trabajos de sus alumnos quedan todos recogidos en ficheros ubicados en el mismo sitio. De esta forma, las respuestas de los alumnos están verificadas automáticamente y mejor organizadas para su corrección por parte del profesor.

Sin embargo, a día de hoy, en las asignaturas de Base de Datos muy pocos grupos utilizan una herramienta que, como los jueces de programación, proporcionan al alumno una diversidad de ejercicios y retroalimentación sobre las respuestas que éste envía, ni mecanismos de motivación o ludificación, muy importantes para captar y mantener la atención y dedicación del alumnado en la asignatura.

Durante el presente curso 2020-2021 se ha realizado una prueba piloto en un grupo de Bases de Datos con la herramienta Learn SQL, en la que se basa este proyecto, que es un sistema de corrección automática de bases de datos, con resultados muy prometedores.

1.2 Objetivos

El sistema Learn SQL forma parte del proyecto de innovación educativa “*Juez automático para el aprendizaje de base de datos”* del programa INNOVA-Docencia de la convocatoria 2020-2021 (número de referencia 18).

Como se ha comentado en el apartado anterior, el objetivo fundamental de este trabajo consiste en ampliar esta herramienta, que es una solución de código abierto y cuya idea es ser usada en la UCM en la asignatura de Bases de Datos, con una serie de mejoras que se comentarán a continuación:

- Mejorar la usabilidad de la aplicación: en el estado inicial, se echa en falta un filtro que seleccione los envíos realizados por el alumno para poder agilizar la

búsqueda de soluciones. También se precisa descargar la solución enviada para ser probada en cualquier otra herramienta. Además, en el caso de tener una clasificación general de la clase, poder generar un Excel donde se guarde la información necesaria.

- Añadir funcionalidades y progresos para aumentar la motivación del alumno y la ludificación: poder generar una vista general, donde alumnos y profesores puedan tener una clasificación de los ejercicios e intentos. Añadir a cada problema un ranking con los tres primeros alumnos en solucionarlo, y otorgar a estos una serie de logros. Así mismo, cuando el alumno se encuentre atascado en un problema, ofrecerle una ayuda para solventar dicho ejercicio.

- Dar soporte en diversos idiomas: capacitar a la herramienta para seleccionar el idioma en el que se muestra la página.

- Habilitar múltiples grupos de clase: analizar y habilitar los modelos que aporta Django de grupos y usuarios para poder usarlos en la aplicación.

- Incorporar nuevos tipos de problemas y mejoras sobre los problemas existentes:
    a) Para los problemas de tipo “SELECT”, poder tener una serie de bases de datos iniciales con distintos valores para poder tener un mayor abanico de datos y dar por válido las soluciones de los alumnos.

    b) La creación de un nuevo tipo de problema donde los alumnos piensen correctamente la solución, ya que aunque el resultado sea correcto para la base de datos inicial, puede ser que no siempre sea la solución correcta.

Las mejoras y nuevas funcionalidades desarrolladas en este trabajo para cumplir con estos objetivos conllevarán una serie de ventajas tanto para los alumnos como para los profesores.

Por parte del alumnado, la principal funcionalidad es el poder tener una aplicación donde poder subir su código y obtener una respuesta inmediata de su corrección.

Los sistemas de ludificación que se proponen en este trabajo aportarán una motivación extra ya que, por una parte, los logros proporcionarán al estudiante un incentivo por conseguir unas metas y las pistas le ayudarán a resolver los ejercicios más complejos cuando el alumno esté atascado y no pueda preguntar al profesor, pues le permiten obtener una ayuda al momento.

Por parte del profesor, la herramienta facilita tener un mayor control de la asignatura, contemplar la evolución de los alumnos a lo largo del curso y poder orientarlos. Su uso lleva consigo albergar las estadísticas del conjunto de ejercicios realizados para años posteriores, permitiéndole mejorar la docencia de la asignatura en los siguientes cursos.

1.3 Plan de Trabajo

Para el desarrollo del proyecto se ha seguido una metodología *Open Source*, ya que los integrantes del equipo de desarrollo no se conocían entre ellos y, debido a las circunstancias ocasionadas por la situación sanitaria durante este curso, todas las reuniones han sido online.

Una vez establecida una lista de propuestas, los miembros del equipo fueron seleccionando las tareas a realizar a lo largo del desarrollo del proyecto. Durante el periodo de desarrollo se realizaron una serie de reuniones periódicas con distinta frecuencia según la fase de desarrollo:

De septiembre a enero, por disponibilidad del equipo de desarrollo, había reuniones cada dos semanas, para ver avances de los mismos y comentarios y revisiones por parte del equipo de revisión, en este caso, los tutores. De enero en adelante, el nivel de trabajo fue mucho más elevado y las reuniones pasaron a ser semanales para una mayor revisión y mejor fluidez del trabajo realizado.

Cada desarrollador ha ido realizando sus partes con la supervisión directa de los tutores, ya que al pertenecer a un proyecto de innovación educativa en el que el código estará disponible públicamente, éste tiene que ser limpio, legible y revisado

para facilitar modificaciones posteriores con las que otros desarrolladores añadan nuevas funcionalidades.

El desarrollo del proyecto se ha realizado en varios marcos temporales, solapado en algunas funciones, donde se contemplan diferentes actividades, como se detalla a continuación:

El primer marco temporal transcurrió en los meses de septiembre a octubre, cuando el equipo de desarrollo se centró en el entendimiento del sistema, cómo funcionaba la aplicación, la lectura de código y la lectura de la documentación...

El segundo marco tuvo lugar entre septiembre y enero. En este tramo, algunas funcionalidades de programación empezaron a llevarse a cabo, allanando el camino para el siguiente marco. Así mismo, fue el punto en el que el equipo empezó a realizar las primeras tareas; en principio las más asequibles, para una mejor adaptación, ya que la curva de aprendizaje era elevada por las herramientas utilizadas.

El tercer marco transcurrió entre febrero y marzo, cuando el nivel de trabajo aumentó considerablemente, ya que la disponibilidad de todos era mayor y la curva de aprendizaje era más asequible. Se realizaron las tareas con mayor dificultad y la finalización del conjunto total de tareas del equipo.

El último marco duró de abril a junio, con la finalización de las tareas por parte de los miembros del equipo, centrándose en la realización de la memoria y revisión de documentación.

2. Introduction

2.1 Motivation

Universities all over Spain include subjects related to programming languages whose assignments, exercises and other practical parts are still done on paper. This means that the correction by the teachers takes longer and, consequently, this feedback is received by the students very late.

Thus, in an attempt to speed up this feedback, the Faculty of Computer Science at the Complutense University of Madrid (UCM) has recently started using programming judges for several programming and algorithmic subjects, such as Fundamentals of Programming, Fundamentals of Algorithmics and Data Structures. However, although with this type of system the student receives immediate feedback on the answer given to the proposed problems, the result obtained by the system is usually quite generic and does not provide much information.

An important aspect of this type of automatic exercise correction applications is the possibility of obtaining statistical results on student activity that can be used to improve teaching in subsequent years. For example, it is possible to obtain the total number of exercises submitted by students, the periods in which most exercises have been submitted, the problems that have required the greatest number of attempts, in which part of the course there is a greater number of students who decide to abandon the subject, etc. This information can be very valuable to reinforce those elements of the course contents that require the greatest effort from the students; in some applications these statistics are visible to the students in the form of a ranking, where they can see their own position in the ranking as well as their classmates. The aim of this system is to motivate them by watching themselves winning "medals" as a reward for having managed to solve a certain number of exercises. This makes this "reward" an incentive to continue with the subject and solve more exercises.

The programming judges have been used in some of the faculty's subjects for exams or assessment tests. The student is programming in a familiar tool; for the teacher, the students' work is all collected in files located in the same place. In this way, the students' answers are automatically verified and better organised for correction by the teacher.

However, in Database subjects there are very few groups that currently use a tool that, like programming judges, provides students with a variety of exercises and feedback on the answers they submit, nor motivation or gamification mechanisms, which are very important for capturing and maintaining the students' attention and dedication to the subject.

During the current academic year 2020-2021, a pilot test has been carried out in a database group with the Learn SQL database correction system, that this project is based on, with very promising results.

2.2 Objectives

The Learn SQL system is part of the educational innovation project "A judge for database learning" of the INNOVA-Docencia programme of the 2020-2021 call for proposals (reference number 18).

As mentioned in the previous section, the main objective of this work is to extend this tool, which is an open source solution and whose idea is to be used at UCM in the subject of Databases, with a series of improvements that will be discussed below:

- Improve the usability of the application: in the initial state, there is a lack of a filter to select the submissions made by the student in order to speed up the search for solutions. It is also necessary to download the submitted solution to be tested in any other tool. Furthermore, in the case of having a general classification of the class, it would be possible to generate an Excel file to store the necessary information.

- Add functionalities and progress to increase student motivation and gamification, such as the following: generation of an overview where students and teachers can have a ranking of exercises and attempts made by students; a ranking of the first three students that have solved each problem; and a collection of awards for the students when they obtain a series of achievements. Also, when the student is stuck on a problem, offer him/her preconfigured help tips to solve the exercise.
- Support in different languages: enable the tool to select the language in which the page is displayed.

- Enable multiple class groups: analyse and enable the models provided by Django for groups and users in order to use them in the application.
- Incorporate new types of problems and improvements to existing problems:

    a) For "SELECT" type problems, to be able to have a series of initial databases with different values in order to have a wider range of data and to take the students' solutions as valid.

    b) The creation of a new type of problem where students think about the solution correctly, as although the result is correct for the initial database, it may not always be the correct solution.

The improvements and new functionalities developed in this work to meet these objectives will entail a series of advantages for both students and teachers.

For the students, the most relevant functionality provided by this tool is to have an application where they can upload their code and get immediate feedback on its correction. The gamification systems proposed in this work will provide extra motivation since, on the one hand, the achievements will provide the student with an incentive to reach certain goals and the hints will help him/her to solve the most complex exercises when the student is stuck and cannot ask the teacher, since they allow him/her to get help immediately.

For the teacher, the tool makes it easier to have greater control of the subject, to see the progress of the students throughout the course and to be able to guide them. Its use produces usage statistics of all the exercises answered by students that can be then analyzed to improve the teaching of the subject in subsequent years.

2.3 Work Plan

An open source methodology was used for the development of the project, as the members of the development team did not know each other and, due to the circumstances caused by the health situation during this academic year, all meetings were held online.

At the beginning of the project a list of proposals was established and the team members selected the tasks to be carried out during the development of the project. During the development period, a series of regular meetings were held with varying frequency depending on the stage of development:

From September to January, due to the availability of the development team, there were meetings every two weeks, to check the progress of the projects and comments and reviews by the review team, in this case, the tutors. From January onwards, the level of work was much higher and the meetings became weekly for a greater review and better flow of the work done.

Each developer has been working on their parts under the direct supervision of the tutors, since, being part of an educational innovation project in which the code will be publicly available, it has to be clean, readable and thoroughly reviewed to facilitate subsequent modifications so that other developers can add new functionalities.

The development of the project has been carried out in several time frames, overlapping in some functions, where different activities are contemplated, as detailed below:

The first timeframe ran from September to October, when the development team focused on understanding the system, how the application worked, studying code and reading documentation.

The second framework took place between September and January. During this period, some programming functionalities started to be implemented, paving the way for the next framework. It was also the point at which the team began to carry out the first tasks; in principle, the most accessible ones, for better adaptation, as the learning curve was steep due to the tools used.

The third frame took place between February and March, when the level of work increased considerably, as all team members were more available and the learning curve was more manageable. The most difficult tasks were completed during this frame.

The last frame lasted from April to June, with the completion of the tasks by the team members, focusing on the completion of the report and review of documentation.

3. Herramientas utilizadas en el proyecto

A lo largo del desarrollo del proyecto se han utilizado muchas herramientas encargadas de facilitar varios aspectos del trabajo, desde la comunicación entre los componentes del equipo hasta la calidad del código.

Este proyecto se ha desarrollado en dos sistemas operativos diferentes: Windows y Linux. En la Facultad de Informática de la Universidad Complutense de Madrid se va a desplegar sobre Linux, en su versión Ubuntu 20.04. Gracias a que los desarrolladores han utilizado distintos sistemas operativos se ha podido comprobar el correcto funcionamiento del juez en ambos sistemas.

A continuación se describen las herramientas utilizadas a lo largo del proyecto.

3.1 Django

Django [1] es un *framework* de desarrollo web de alto nivel escrito en Python. Es de código abierto. Está creado para ayudar a los desarrolladores a crear aplicaciones web grandes de una manera más sencilla.

Es un *framework* basado en plantillas y en la reutilización de código. Se utilizó debido a su gran potencia y sencillez.

3.2 Oracle y PostgreSQL

Se utilizaron dos gestores de bases de datos, Oracle y PostgreSQL. Oracle [2] es un sistema de gestión de bases de datos de tipo relacional. Se ha utilizado Oracle ya que en la Facultad de Informática es la herramienta utilizada para impartir las asignaturas relacionadas con bases de datos. Además, dentro de Learn SQL se ha utilizado para ejecutar los envíos que realizan los alumnos.

PostgreSQL [3] es un sistema de gestión de bases de datos de tipo relacional, se caracteriza por ser de código abierto. Se ha utilizado a través del ORM desde Django para mantener la información interna de la aplicación. Un ORM es un

modelo de programación que permite mapear los objetos utilizados por un programa a una base de datos relacional para obtener persistencia. Se ha decidido mantener los dos gestores de bases de datos en la aplicación para cumplir con las necesidades de las asignaturas.

3.3 Pylint y Codecov

El código fuente escrito en este trabajo se ha escrito por varios desarrolladores, y además está previsto que otros participantes amplíen el sistema una vez finalizado este proyecto. Como mantener un código uniforme y entendible no es sencillo, para poder asegurar que se aplican buenas prácticas de programación se ha decidido utilizar Pylint.

Pylint [4] es una herramienta que analiza el código fuente en busca de errores en la nomenclatura, espacios sobrantes o realiza sugerencias de refactorización. Esta herramienta proporciona una puntuación basándose en el cumplimiento de estos criterios de calidad, la escala de valoración proporciona una calificación entre 0 y 10, siendo 10 la nota más alta. En nuestro proyecto se ha establecido el nivel más alto de exigencia, por lo que debe obtenerse la máxima nota para poder integrar las modificaciones realizadas por los programadores.

Además otra buena práctica de desarrollo de aplicaciones consiste en realizar test automáticos que comprueben el correcto funcionamiento de todas las implementaciones creadas, y también para comprobar que con las nuevas implementaciones no han dejado de funcionar implementaciones creadas con anterioridad.

Con el objetivo de automatizar la realización de test se ha utilizado la herramienta Codecov [5], una solución de cobertura de código. En este trabajo se ha utilizado para comprobar qué porcentaje de líneas de código cubrían los test creados. Como con Pylint, en nuestro proyecto se ha establecido que el criterio de aceptación debe ser el máximo, por tanto se ha establecido para los test una cobertura del 100% de líneas de código.

3.4 Editores de texto

Para llevar a cabo el desarrollo de la aplicación se han utilizado varios editores de texto.

Pycharm [6] es un IDE de Python que impulsa la calidad del código. Se utilizó para la mayoría del desarrollo del código, tanto para Python como para Javascript y HTML.

Notepad++ es un editor de texto que soporta varios lenguajes [7]. En nuestro caso se utilizó para leer archivos `JSON`, los cuales se utilizan en la configuración de problemas nuevos introducidos por archivos comprimidos.

Sublime Text 3 tiene un sistema de colores muy marcado que facilita la lectura de sentencias SQL [8], se ha utilizado para crear y leer de una manera más sencilla sentencias SQL para poder utilizarla en los problemas.

3.5 Herramientas de comunicación y gestión de proyecto

Uno de los principales aspectos a tratar en un proyecto de programación con varios desarrolladores independientes es el compartir el código fuente del proyecto. Para esto y, además, controlar los cambios, se utilizó el sistema GitHub, que actualmente es la herramienta más utilizada del mundo para desarrollos de sistemas en equipo. Además, GitHub está especialmente diseñada para el desarrollo de aplicaciones de código abierto. En particular, se ha utilizado el desarrollo por ramas que nos ofrece git [9] a través de GitHub. El proyecto se almacena en la rama principal, denominada *master*, mientras el resto de ramas son copias (clones) de esta rama principal, para que cada desarrollador pueda introducir los cambios de sus implementaciones sin cambiar el código de la rama principal, para evitar conflictos con los cambios realizados por los demás desarrolladores.

Respecto a la comunicación entre los desarrolladores, a lo largo del proyecto se han hecho reuniones semanales para llevar una comunicación continuada entre los componentes del equipo y los supervisores. Las reuniones han tenido que ser remotas por la situación sanitaria durante el curso, para poder planificar estas reuniones y llevarlas a cabo se han utilizado las siguientes herramientas.

WhenIsGood (https://whenisgood.net/) es un planificador online para poder marcar los horarios disponibles de cada uno de los integrantes del grupo y de los tutores. Con esta herramienta concretamos las reuniones durante el primer cuatrimestre del curso.

A partir de febrero pasamos a Doodle (https://doodle.com/es/) ya que es una herramienta más potente con la misma funcionalidad que WhenIsGood, pero en este caso además lo pudimos asociar al correo electrónico de cada uno de los invitados a la reunión para que así nos llegara un aviso con el enlace a la reunión.

Las reuniones se llevaron a cabo en Meet, una aplicación de videoconferencia, donde se puede compartir pantalla. Esta herramienta fue muy útil para compartir algunas de las dudas y poder mostrar a los tutores las partes del código donde pudiéramos necesitar ayuda.

Como las reuniones se han realizado con frecuencia semanal, era necesaria otra vía de comunicación con los tutores para el momento en el que surgían dudas. En las primeras tareas esta comunicación se ha realizado a través de Gmail, pero este intercambio de mensajes era un poco lento y el incremento en el número de mensajes hacía difícil la comunicación y colaboración. Por este motivo, más adelante se decidió utilizar dos de los componentes más potentes de GitHub, los *Issues* y los *Pull Request*.

En los *Issues* se marcan los errores, fallos detectados o implementaciones que se quieren añadir al sistema. En nuestro caso los utilizábamos para agregar las tareas de cada componente del equipo de desarrollo. Los *Pull Request* se utilizan cuando el código ya está implementado en nuestra rama local y se quieren integrar los cambios en la rama principal donde están todas las mejoras implementadas; aquí el

administrador de la rama principal también puede añadir anotaciones, en las que puede comentar mejoras del código que estamos queriendo agregar a la rama principal. Esta práctica de comunicación en los equipos de trabajo está muy extendida entre los desarrolladores de código libre.

3.6 Diseño de bocetos

Durante el desarrollo de aplicaciones es necesario tener herramientas sencillas para hacer diseños rápidos de bocetos que muestren de forma esquemática el aspecto de la interfaz de usuario para poder presentarlo a los demás componentes del equipo de desarrollo y facilitar el acuerdo entre los miembros del grupo sobre cómo se debían mostrar algunas vistas. Estos bocetos se denominan *mockups* en la terminología del desarrollo de aplicaciones. En este trabajo se han utilizado distintas herramientas que se describen a continuación.

Balsamiq [10] es una aplicación para crear diseños de páginas web que se utilizó para crear el *mockup* de la vista de los resultados. Esta aplicación es de pago, con posibilidad de un mes de prueba gratuita, por eso solo se utilizó en un caso.

Frame Box [11] es una herramienta gratuita donde se pueden crear *mockups* de manera rápida y online que se utilizó para crear el *mockup* de las vistas de los logros. Poco después de realizar este *mockup* esta página desapareció de la web.

Por último, Paint [12] es un editor de imágenes de Microsoft que se utilizó en la creación del *mockup* de las pistas ya que se debía editar una imagen y no crear un *mockup* desde cero.

4. Plataformas antecesoras

4.1 Jueces y sistemas automáticos de evaluación

El término de "juez de programación" proviene de los concursos de programación: es un sistema automático que verifica que el programa entregado por los grupos que compiten en el concurso funciona correctamente con un conjunto de casos de prueba normalmente oculto.

El uso de jueces automáticos para la evaluación de ejercicios en línea es muy útil para los profesores. La gran cantidad de alumnos en los grupos de las asignaturas de programación y de bases de datos hace que la tarea de corregir múltiples ejercicios a cada uno de los alumnos sea muy laboriosa y los resultados de las correcciones lleguen a los alumnos muchos días o semanas después de haberlos realizado. Además los resultados obtenidos en el juez se pueden utilizar directa o indirectamente como un elemento más de la evaluación de los alumnos. Asimismo, para los alumnos también resultan de gran utilidad: pueden ir progresando con los ejercicios de la asignatura a su ritmo, y cuando realizan estos ejercicios obtienen un resultado inmediato, por lo que pueden utilizar este resultado para volver a repasar los conceptos que permiten desarrollar el ejercicio. Además, como los resultados de los alumnos suelen ser públicos, se añade un factor de competitividad y desafío que anima a ejercitarse más en la asignatura.

Para la realización de este trabajo se han evaluado otros sistemas existentes para la evaluación de ejercicios. En particular, los sistemas que se han tenido como referencia han sido DomJudge, Acepta el Reto y FLOP*,* entre otros.

A lo largo de la carrera se han utilizado este tipo de sistemas en diferentes asignaturas como Fundamentos de la Algoritmia, Estructura de Datos, Diseño de Algoritmos o Técnicas Algorítmicas en Ingeniería de Software. Nuestro juez, Learn SQL, es un juez desarrollado para la asignatura de Bases de Datos. En esta sección se examinarán las diferencias fundamentales de los sistemas existentes con Learn SQL.

DomJudge [13] (https://www.domjudge.org/) es un juez de programación configurable en C, C++ y Java. Se utiliza para los concursos ACM/ICPC (*International Collegiate Programming Contest*).

Acepta el Reto (https://www.aceptaelreto.com/) es un juez en línea de problemas de programación en español desarrollado en la Facultad de Informática de la UCM que acepta soluciones en C, C++ y Java. Se utiliza en asignaturas como: Fundamentos de la Algoritmia, Estructura de Datos, Diseño de Algoritmos o Técnicas algorítmicas en Ingeniería de Software, además de usarse en entrenamientos de grupos para concursos de programación del tipo de ACM/ICPC. Acepta el Reto tiene una colección de más de 500 problemas y permite evaluación de la complejidad en tiempo y en espacio en memoria.

Por último, FLOP [14] (http://problem-g.estad.ucm.es/FLOP/index.jsp) es un laboratorio virtual para practicar la programación . Contiene multitud de problemas que se pueden resolver en C++, Java, Python, Pascal o Haskell. Se utiliza en asignaturas de la Facultad de Ciencias Matemáticas. Es la única plataforma que puedes usar sin necesidad de tener que estar identificado.

Las plataformas estudiadas son bastante básicas a la hora dar una retroalimentación al alumno de los ejercicios enviados. Solo emite unos pocos resultados posibles: error de compilación, error de ejecución, resultado incorrecto, resultado correcto. Además, lo habitual en los jueces automáticos es mantener un conjunto de casos de prueba que nunca se muestran al participante en el concurso. Por este motivo, muchas veces no se entiende exactamente qué es lo que está fallando, por lo que en algunos casos no son tan útiles en las primeras fases del aprendizaje de un nuevo lenguaje de programación.

A continuación se detallan algunas de las características más relevantes de estos sistemas que los diferencian de Learn SQL.

La diferencia más relevante entre los sistemas descritos anteriormente y el nuestro es que esos sistemas anteriores están enfocados en la programación de diferentes lenguajes y nuestro sistema está orientado a las bases de datos.

Otra diferencia significativa con nuestro sistema la presenta DomJudge, pues cuenta con un sistema de mensajería para que el alumno pueda intercambiar mensajes con el profesor, funcionalidad inexistente en Learn SQL. Sin embargo, cuando un alumno tiene un problema y necesita ayuda, depende de la disponibilidad del profesor para conectarse y atender todas las peticiones de los alumnos. Por este motivo, en Learn SQL se ha desarrollado un sistema de pistas para que esa petición de ayuda al profesor sea automática y por lo tanto mucho más ágil. Así el alumno no tiene que esperar una respuesta para poder obtener ayuda con algún ejercicio, sino que puede obtener automática e inmediatamente pistas preconfiguradas por el profesor sobre la forma de enfocar la resolución de un problema. Este sistema, además de ofrecer ventajas evidentes para el alumno, permite al profesor establecer ayudas sobre determinados ejercicios sin necesidad de contestar una y otra vez a los alumnos que van consultando dudas en el sistema de mensajería.

Por otra parte, varios sistemas estudiados permiten obtener información estadística de envíos realizados por alumnos: tanto DomJudge como Acepta el Reto cuentan con una ventana para ver las estadísticas de los usuarios en cuanto a envíos; en DomJudge se puede ver una tabla con los envíos realizados por los demás alumnos y en Acepta el Reto puedes ver un listado con los envíos efectuados por los usuarios.

Nuestro juez cuenta con una pestaña de clasificación que contiene una tabla a modo de ranking con toda la información de los alumnos de un grupo para una colección en concreto. A diferencia de los demás jueces, esta clasificación se puede filtrar por grupo y por fechas.

Respecto al soporte de múltiples idiomas, tanto FLOP como nuestro juez cuentan con soporte multidioma, español e inglés, a diferencia del resto de los sistemas estudiados.

En cuanto al fomento de la motivación de los alumnos para resolver los problemas propuestos y aumentar la competitividad, en los sistemas estudiados se ha observado que tan solo disponen de un ranking en el que aparece una lista de los

alumnos con un resumen del número y resultados de sus envíos. En el sistema Learn SQL se ha ido un paso más allá, ofreciendo un sistema de logros y de podio con los tres primeros alumnos en resolver cada ejercicio.

Otra diferencia que existe entre los distintos sistemas y el nuestro es que Learn SQL al ser un sistema de bases de datos, en los que la ejecución y pruebas de la solución del problema requieren el uso de un esquema de base de datos que en otros contextos de programación es innecesario, incluye la funcionalidad de poder descargar los archivos del juez.

Todos estos sistemas descritos al igual que Learn SQL son de código abierto y gratuito.

4.2 Herramientas para el aprendizaje de SQL

Los lenguajes utilizados en los sistemas de bases de datos, especialmente SQL y otros lenguajes de consulta de bases de datos, tienen características específicas que los distinguen de los lenguajes de programación que son utilizados en los jueces de programación vistos en el apartado anterior. Una diferencia importante es que se han desarrollado para el modelo relacional de datos, que está fundamentado en un modelo teórico basado en la teoría de conjuntos. Esto hace que los lenguajes de consulta tengan una gran potencia expresiva, de forma que pueden resolver consultas muy complejas de manera muy concisa. El inconveniente fundamental de este tipo de lenguajes es que no es fácil dominarlos y la curva de aprendizaje tiende a ser más larga. Por ello, para las asignaturas de Bases de Datos son muy útiles las herramientas para el aprendizaje de SQL como w3school, Live SQL, SQL ZOO, SQL Fiddle, SQL Bolt o DES (*Datalog Educational System*).

Estas herramientas sirven para que los alumnos practiquen ejercicios para la asignatura. Algunas de ellas como w3school [15], SQL ZOO [17] o SQL Bolt [19] son páginas dedicadas a ejercicios interactivos de SQL, con ejemplos de todo tipo de problemas con grados de dificultad crecientes. En todos los ejercicios te muestran la tabla resultante con la información que debería devolver el resultado del código

enviado. En cambio las demás, Live SQL [16], SQL Fiddle [18] o DES [20], además de tener ejemplos de ejercicios y practicar con ellos, puedes crearte tu propio esquema con tus tablas y datos, y realizar las consultas que te pidan en el enunciado de tus ejercicios.

Estas herramientas se parecen mucho a los jueces automáticos en cuanto al *feedback* con los resultados esperados de las consultas o con los errores que ha generado el código. Pero a su vez carecen de algunas características de los mencionados como es esa parte de competición sana al poder ver las estadísticas de los demás usuarios, una clasificación de los usuarios para ver en qué puesto estás en comparación con los demás, un sistema de logros que sirva de motivación e incentivo para seguir haciendo más ejercicios y así obtener esa “recompensa”.

En definitiva, Learn SQL al combinar la competitividad característica de los jueces de programación y las ayudas particulares de los sistemas de aprendizaje de SQL, se puede considerar la herramienta perfecta.

5. Estado inicial de Learn SQL

Learn SQL ya contaba con una buena base lista para ser utilizada, pero la motivación que nos llevó a realizar este trabajo fue la cantidad de extensiones que podrían ser añadidas para mejorarla. En este capítulo se introducen los elementos más importantes de la versión inicial de Learn SQL en la que se basa este Trabajo de Fin de Grado (TFG).

Por una parte, al inicio de este trabajo Learn SQL contaba con un sistema de usuarios en el cual se podían diferenciar dos tipos de usuario: los usuarios sin permisos y los administradores. Ambos tipos de usuarios debían estar registrados e iniciar sesión para poder realizar todas sus funciones. Los usuarios sin permisos podían acceder a todo el contenido relacionado con visualizar los problemas, consultar sus envíos, resolver ejercicios y cambiar su contraseña. Por otro lado, los usuarios administradores contaban con los mismos permisos que los usuarios anteriormente descritos y además la posibilidad de acceder a la vista de administrador, donde se podían crear las colecciones, los problemas y los usuarios y cambiar sus datos así como los permisos, y subir ejercicios y colecciones enteras tanto de manera manual como a partir de archivos `ZIP`. En este aspecto, el sistema claramente necesitaba incluir mecanismos de agrupación de usuarios para poder utilizarlo de forma práctica, por lo que se añadió la investigación y documentación sobre la creación de grupos y la asignación de los usuarios a dichos grupos, para poder incluir, por ejemplo, alumnos y profesores en un mismo grupo y así distinguir entre diferentes clases.

En la versión inicial del sistema, Learn SQL contenía los problemas guardados en grupos llamados colecciones. En estas colecciones los usuarios podían ver información sobre el número de problemas que ya se han resuelto y el número total de problemas que poseía esa colección. Además, dentro de cada colección se podían ver los problemas incluidos junto con el número de envíos que habían sido realizados por el usuario.

Learn SQL contaba con varios tipos de problemas: problemas de consultas SQL (“SELECT”), problemas de disparadores, problemas DML, problemas de definición de funciones y problemas de procedimientos. En este apartado del sistema, se vio una clara oportunidad de extender el modelo añadiendo nuevos tipos de problemas para su mejora en cuanto a la cantidad de tipos de contenido evaluable.

En cada problema planteado a los usuarios se distinguían varios elementos:

- El enunciado, donde se muestra una explicación del problema.

- La descripción del esquema de la base de datos inicial sobre la que se debía ejecutar el código de alumno, junto con datos iniciales para entender el contenido de la base de datos y el objetivo del problema.

- El resultado esperado, el cual contenía los resultados que debería obtener una solución para considerarse como correcta sobre la base de datos con los valores iniciales mostrados en el enunciado.

- El cuadro donde se introducía el código SQL de la solución, junto con el botón “Enviar solución”, el cual enviaba al juez la solución que había en el cuadro y se corregía. El juez podía contestar con distintos veredictos posibles: aceptado, error en ejecución, resultados incorrectos, error de validación, tiempo límite excedido o error interno.

En este trabajo se ha mantenido esta estructura básica de los ejercicios, aunque se han propuesto muchas mejoras para facilitar su uso por parte de los alumnos. Por ejemplo, algunas de las mejoras realizadas consisten en, añadir un botón para poder descargar un archivo `SQL` con las sentencias para la creación de la base de datos, un botón para reducir y ampliar el contenido de las tablas, un botón para subir directamente el archivo `SQL` como solución sin tener que escribir el contenido directamente en el cuadro de código de la solución, etc.

Una vez enviada una solución, el sistema podía devolver varios tipos de resultados:

- *AC (Accepted)*, respuesta correcta.

- *WA (Wrong Answer)*, respuesta incorrecta. Este tipo de respuesta venía acompañada de una retroalimentación sobre los fallos en la solución proporcionada.

- *RE (Runtime Error)*, error de ejecución.

- *TLE (Time Limit Error)*, tiempo límite excedido.

- *IE (Internal Error)*, error interno.

- *VE (Validation Error)*, error de validación.

En la versión inicial, las respuestas del sistema eran bastante básicas, lo que no facilitaba al alumno descubrir los errores que tenía su solución al problema propuesto, por lo que una parte de las tareas realizadas en este trabajo ha consistido en mejorar la retroalimentación al usuario. Para ello se han realizado diferentes tipos de mejoras que proporcionen una mejor respuesta por parte del sistema, como por ejemplo, frases más concisas y objetivas sobre los fallos y errores cometidos como el número de filas de más que ha dado como resultado la solución, la incorporación de más de una base de datos para la corrección de los problemas, de tal manera que se pueda especificar de manera más precisa la solución objetivo y así especificar de mejor los posibles fallos.

Una de las principales carencias del sistema, en la cual se han centrado la mayoría de las nuevas implementaciones a la hora de extender sus funcionalidades, fue en los mecanismos de motivación al estudiante para la realización de los problemas y en la sensación de no ser la única persona que está realizando ejercicios. En este último aspecto, Learn SQL no contaba con ningún tipo de estadística ni indicio sobre el número de usuarios que había realizado un problema concreto, o el número de usuarios enviando soluciones. Se sentía que los usuarios estaban completamente aislados unos de otros. Para ello, se llevó a cabo la implementación de un sistema de clasificación en el que se podía ver, para cada colección, una tabla con las estadísticas de todos los problemas y alumnos de dicha colección. De esta manera los alumnos ya se podrían ver más conectados y relacionados con el resto de usuarios.

En cuanto a los mecanismos para la motivación, se vió necesaria la implementación de la obtención de logros al llevar a cabo diferentes hazañas en el sistema, un podio con el “Top 3” de los usuarios en cada problema, un sistema de pistas para intentar evitar que los alumnos se rindieran al no poder completar los problemas.

6. Metodología de desarrollo

Por motivos de la COVID-19*,* el desarrollo del proyecto desde el primer instante se planteó que fuera 100% online, ya que las clases en su mayoría se impartieron por medios digitales. Además, el equipo del proyecto estaba planteado desde el principio como una colaboración entre estudiantes que no se conocían entre sí y no había un trabajo previo juntos y este hecho marcó una serie de pautas para tener un trabajo mejor controlado y tutelado por los tutores.

Al formar parte de un proyecto de innovación docente se utilizó la metodología de desarrollo digital *Open Source*, en la que cualquier persona con conocimientos puede acceder al código fuente y modificarlo, así cualquier centro educativo podría descargar e instalar esta herramienta y plantear sus propias mejoras para la herramienta.

Con la metodología *Open Source* se garantiza la accesibilidad de todas las personas al ser una manera de desarrollo que facilita el código fuente de manera gratuita. También permite la personalización de la herramienta, ya que se puede modificar el código todas las veces que se quiera, no necesariamente después deben ser implementadas en la rama principal del proyecto.

El desarrollo del proyecto se organizó en diferentes nuevas funcionalidades y mejoras, las cuales resultaban más interesantes para enriquecer la herramienta. A partir de esta lista de mejoras, se plantearon las tareas a realizar en el proyecto con distintos niveles de dificultad. Las tareas se dividieron en subtareas.

Para tener el trabajo mejor dividido, se realizó la división de tareas en fragmentos lo más pequeños posibles y también por dificultades (baja, media, alta*).* Esta división hizo posible un reparto más equitativo de tareas en los integrantes del grupo para que el trabajo fuera lo más homogéneo posible.

Inicialmente la comunicación entre los miembros del equipo y los tutores del proyecto fue mediante correo electrónico. Tras trabajar unos meses de esta forma observamos que llevar la cuenta de tantos mensajes tenía un alto nivel de

complejidad, por ello decidimos cambiar el método de comunicación y usar GitHub para este intercambio de mensajes mediante *Issues*. Gracias a los *Issues* teníamos más control sobre los mensajes y un acceso más cómodo a la hora de hacer alguna consulta o revisar los comentarios.

Además de esta comunicación mediante *Issues*, cuando se terminaba una tarea y se hacía un *Pull Request* para integrar la nueva funcionalidad, se mantenía un intercambio de mensajes en el propio *Pull Request* con cambios en el código o mejoras que se encontraban. Un *Pull Request* es una petición de uno de los desarrolladores para incluir sus mejoras en la rama principal, donde se encontraba el proyecto con todas las mejoras que se han ido haciendo. Asimismo, gracias al uso de esta herramienta también se podían controlar los conflictos que surgían entre las tareas de los miembros del equipo.

En el transcurso del proyecto, algunos integrantes del equipo de desarrollo tuvieron conflictos entre sus tareas, que se producían al modificar los mismos archivos durante el desarrollo de sus tareas. Estos conflictos se pudieron ver gracias al uso de la herramienta GitHub y sus funcionalidades, sin ella podríamos haber perdido mucha información, mejoras implementadas por los desarrolladores y tiempo de estos.

Para poder realizar un seguimiento del trabajo, durante el transcurso del TFG se fueron programando reuniones, en las que se organizaban y se comentaba el progreso que llevábamos cada uno sobre nuestras tareas, explicando los problemas y dificultades que nos surgían, y así, poder buscar soluciones entre todos. En caso de acabar las tareas que se nos habían asignado, elegíamos nuestra siguiente tarea para poder comenzar con ella.

Se ha utilizado un sistema de integración contínua, TravisCI, en la que cada una de las implementaciones realizadas por los alumnos se ha ido subiendo a un repositorio de GitHub principal. Por tanto este repositorio se ha ido actualizando constantemente con la última versión del proyecto. Esto ha agilizado el desarrollo, ya que ha permitido que se implementaran mejoras menores e integrarlas en el proyecto

principal. Sin esta metodología de trabajo se debería integrar todas las partes del proyecto de cada uno de los alumnos al final y las colisiones entre estos habrían sido muy grandes, lo que nos habría hecho perder mucho tiempo en el proceso de integración.

También, para mantener la cohesión del código, se ha utilizado una metodología de pruebas con test unitarios, haciendo que cada implementación del código tuviera que pasar todos los test creados con anterioridad y los nuevos creados que hicieran que se cubriera el 100% del código.

7. Implementaciones

En esta sección se explicarán detalladamente las mejoras y extensiones realizadas en el sistema por el equipo de desarrollo. Cada implementación contendrá enlaces a sus *Pull Requests* e *Issues* (si tienen I*ssues* asociados).

7.1 Botón para descargar script de tablas

Enlace Pull Request : https://github.com/emartinm/lsql/pull/15

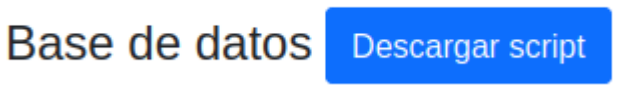
7.1.1 Descripción y motivación

El objetivo de esta tarea consiste en crear un botón en cada problema del sistema para poder generar un archivo `SQL` con las sentencias necesarias para la creación de una base de datos inicial.

En la versión inicial de la aplicación no existe ninguna manera de poder descargarse un *script* con las sentencias SQL de creación de tablas e inserción de los datos de los ejercicios para que el alumno pueda realizar pruebas en su ordenador.

Durante el transcurso de la asignatura se recomienda al alumno que trabaje en su ordenador con una base de datos local para la realización de los ejercicios. Para resolver cada ejercicio, se tiene que crear una base de datos inicial, lo cual a la larga se hace repetitivo.

Sería interesante que existiera una opción en donde hubiera un botón, como el que observamos en la siguiente imagen, que esté dentro de cada problema y que genere un archivo con extensión `SQL` que contenga las sentencias necesarias para la creación de las tablas y la inserción de datos del problema.



La idea surgió debido a que muchos alumnos pueden tener en sus ordenadores varias herramientas para bases de datos. Con esta opción, el alumno puede tener las tablas y los datos de prueba del ejercicio en una herramienta ajena a Learn SQL para ejecutar la sentencia y posteriormente subir la solución una vez resuelta en su ordenador.

7.1.2 Desarrollo

Las primeras interacciones con la tarea se han centrado en buscar documentación relevante sobre el uso de Django [21] para ver la estructura que se usa para el acceso a archivos y rutas de acceso locales, así como el funcionamiento de las llamadas a las funciones de Django para tratarlas.

Una vez se ha comprendido el funcionamiento de estas características de Django, el desarrollo ha consistido en la creación de un botón en las páginas de problemas. Al pulsar sobre él, se creará un fichero `SQL` con las sentencias de creación de las tablas y la inserción de datos que será descargado en el ordenador del alumno.

Durante el desarrollo, hubo dificultades al no poder establecer la extensión `SQL` en el nombre de archivo generado. La solución consistió en definir la propiedad de la cabecera (*Content-Type*) con el tipo "*application/sql*" [22]. De esta forma, las descargas de los archivos serán con extensión `SQL`.

Durante el transcurso de la tarea se realizaron un total de 66 intercambios de mensajes, repartidos entre: 50 por correo y 16 en los *Pull Request*.

7.1.3 Test Unitarios

Para esta tarea, se han creado test unitarios que permiten verificar el uso del botón.

Al realizar la descarga, se verifica que el archivo contiene la creación de tablas y las inserciones correspondientes del problema seleccionado. Además se verifica que la propiedad de la cabecera corresponde a un archivo `SQL` y que el código de respuesta devuelto por la aplicación corresponde a un código de respuesta satisfactorio (200).

Durante la ejecución de los test se realizan un total de 4 comprobaciones.

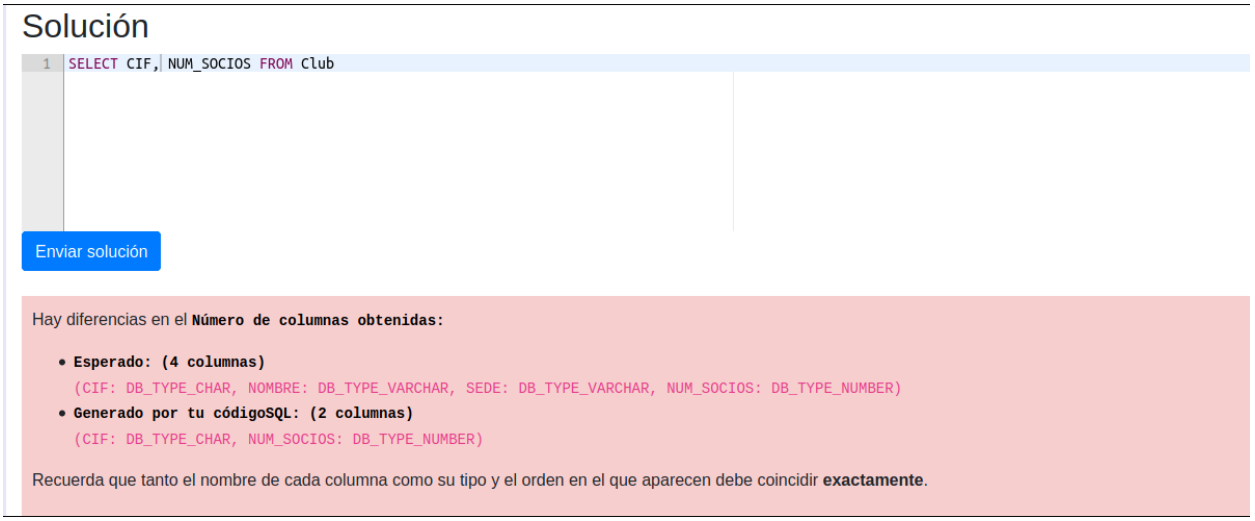
7.2 Mejora de la respuesta generada ante una solución incorrecta en el esquema generado

Enlace Pull Request: https://github.com/emartinm/lsql/pull/19

7.2.1 Descripción y motivación

Cuando el alumno envía las solución de un ejercicio recibe siempre una respuesta, pero la respuesta generada por el sistema es genérica y no proporciona gran información sobre el fallo. Esto puede desmotivar al alumno ya que, al no tener claro qué le ha fallado, hay posibilidades de que abandone el ejercicio y pase a otro.

En la versión inicial de Learn SQL, a la hora de realizar envíos se pueden generar varios veredictos: *RE*, *WA*, *AC*. En el primer caso se lanza una excepción. Entre *WA* y *AC* se realiza una comprobación del esquema de la tabla y la igualdad de las columnas como se puede observar en la siguiente imagen.



El mensaje que se devuelve en caso de ser *WA* es escueto y da poca información al alumno del error. El objetivo de la tarea trata de mejorar ese mensaje de retroalimentación para dar una información más clara del error.

7.2.2 Desarrollo

La tarea trata de mejorar la funcionalidad que realiza la comprobación con el esquema de la tabla. Se distinguen los casos de fallo y acierto y se ha dividido en los siguientes pasos:

1. Si el envío es exactamente igual al esquema de tabla y el número de filas enviará un *AC*.
2. Si en el envío el número de filas es distinto al que debería ser enviará un *WA* con un mensaje erróneo correspondiente al número de columnas.
3. Si en el envío el número de filas es el mismo se recorre columna a columna:
    a. Si los nombres son distintos se enviará un *WA* con un mensaje erróneo correspondiente al nombre de las columnas.
    b. Si el nombre es igual pero son de distintos tipos se enviará un *WA* con un mensaje erróneo correspondiente al tipo de las columnas.

Durante el desarrollo de las pruebas unitarias, se encontró un problema. Los nombres coincidían pero se diferenciaban en minúsculas y mayúsculas y se mostraba un mensaje de error. La solución, en este caso, era correcta, ya que Oracle no distingue entre mayúsculas y minúsculas.

También se mejoró el mensaje de error dado por el nombre y el tipo. Ofrece al estudiante la diferencia entre columnas, el nombre generado por el código del alumno y el nombre esperado.

Durante el transcurso de la tarea se realizaron un total de 56 intercambios de mensajes, repartidos entre: 20 por correo y 36 en el *Pull Request* 19.

7.2.3 Test Unitarios

Para esta tarea, se han creado test unitarios basándose en el envío de una solución de un problema en concreto validando su salida. Las primeras pruebas se basaron en entregas de una solución correcta. Las siguientes pruebas se basaron en veredictos

WA y comprobando que los mensajes de retroalimentación mostraban el número de columnas cuando no coinciden, los nombres o el tipo son distintos al esperado.

Durante la ejecución de los test se realizan un total de 30 comprobaciones.

7.3 Grupos de clase

Enlace Pull Request: Esta tarea no tiene asociado ningún *Pull Request* ya que su intención era la investigación de cómo Django proporciona soporte para los usuarios y los grupos para poder adaptarlos a la aplicación.

7.3.1 Descripción y motivación

En cualquier aplicación existen diversos usuarios, ya sean administradores o usuarios comunes. Learn SQL también distingue entre dos tipos de usuarios, alumno y profesor. El objetivo de esta tarea consiste en poder agrupar a los usuarios en los grupos de clases a los que pertenecen.

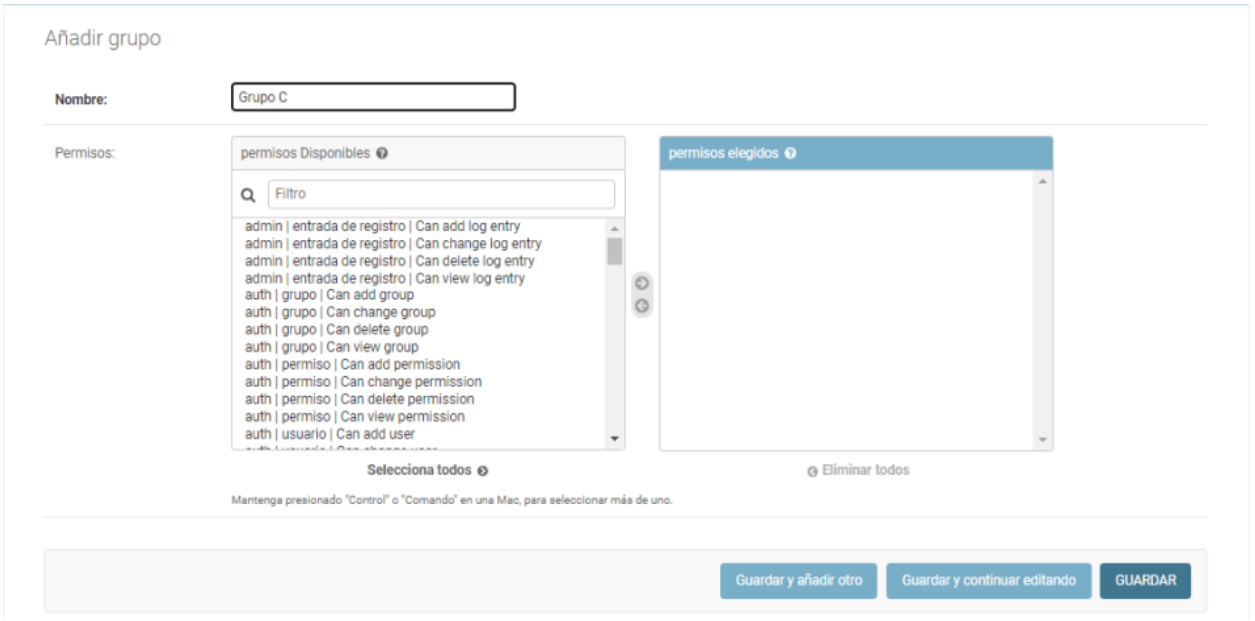
Django ofrece modelos para grupos y usuarios y se ha decidido usar estos modelos predefinidos para definirlos como clases y usuarios (alumnos y profesores).

7.3.2 Desarrollo

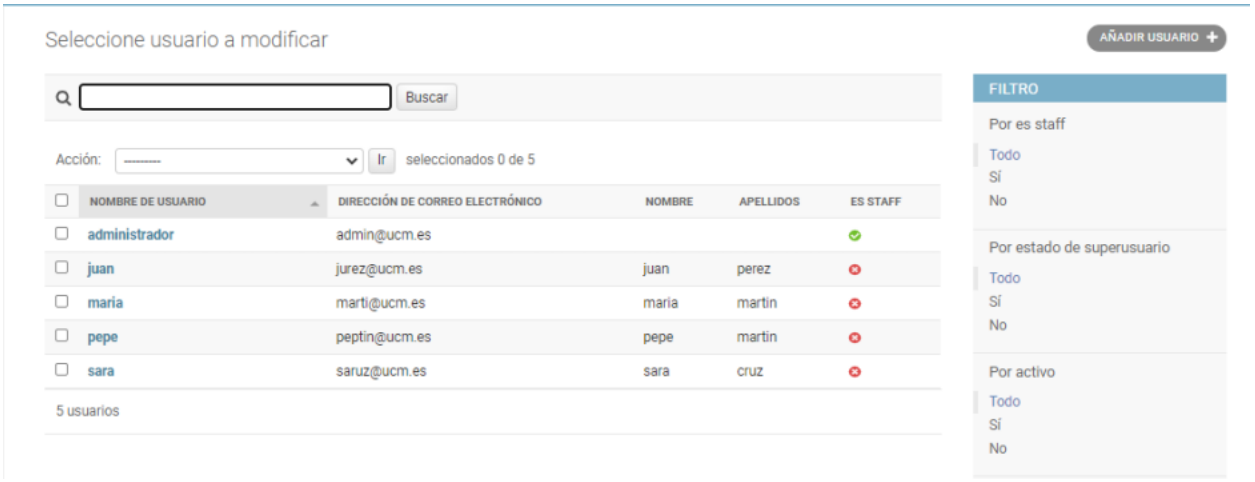
Django ofrece una serie de modelos, grupos y usuarios [23], que se usarán en la aplicación como usuarios, ya sea alumno o profesor, y los grupos, los cuales como objetivo tiene asignar permisos y los usuarios, que se usará para distinguir las clases.

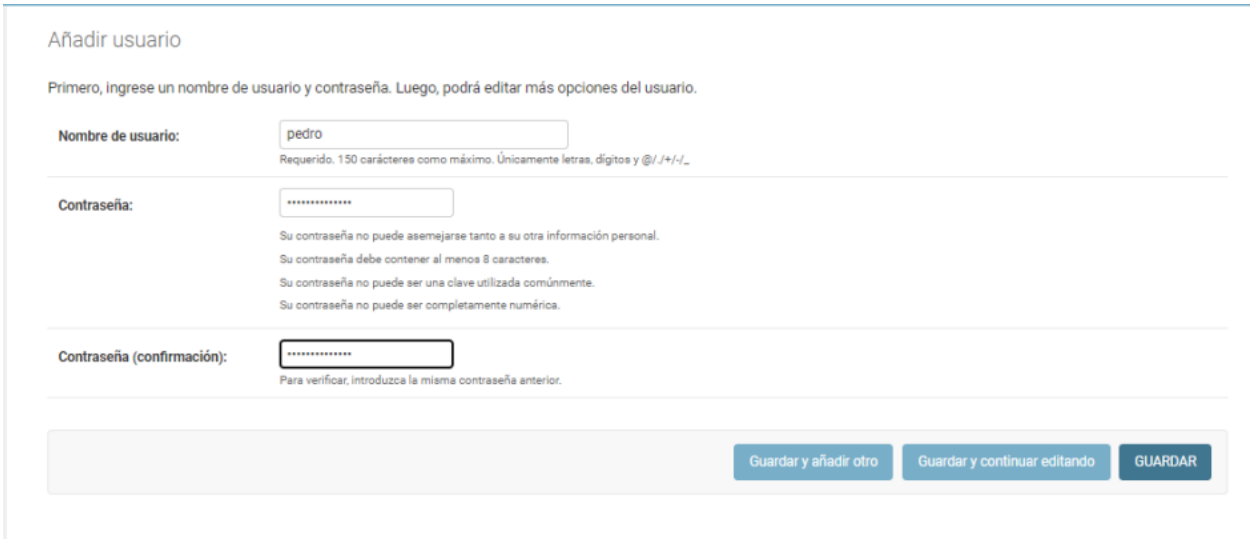
Por parte del profesor, desde la parte de la administración de Django, se pueden observar los modelos que se encuentran en la herramienta, entre ellos los anteriormente mencionados, como se puede observar en la siguiente imagen.

Estos modelos ofrecen un gran abanico de posibilidades para poder organizar sin problemas los grupos y a los usuarios. Para la creación de un grupo, se pulsará en el botón "Añadir grupo" , como se observa en la imagen, y se ingresa el nombre del grupo, en este caso “Grupo C”. Por defecto no tendrán permisos los grupos.



Al tener el grupo creado, se crean los usuarios. Desde la misma vista al crear un grupo se pulsa en el botón "Añadir usuario" y añadimos el nombre del usuario y su contraseña.





En la sección de permisos al crear un usuario, es donde se da la distinción entre alumnos y profesores. Un alumno tendrá permiso *Active*, como se puede observar en el siguiente ejemplo.

Un profesor tendrá permisos *Staff Status.* En la sección de grupos, se elige el grupo anteriormente creado y se presiona la flecha apuntando a la derecha entre las cajas,

como se puede observar en la imagen. Para salvar tanto un grupo como un usuario, simplemente pulsando el botón "guardar".

7.3.3 Test Unitarios

Esta tarea no contiene pruebas unitarias ya que su labor fue de investigación del uso de los modelos de grupos y usuarios que Django ofrece.

7.4 Vista Resultados

Enlace Pull Request: https://github.com/emartinm/lsql/pull/26

7.4.1 Descripción y motivación

Esta tarea consiste en incluir una pantalla que resuma toda la información de la clase y de todos alumnos con los ejercicios realizados y con una clasificación que depende del número de ejercicios e intentos de cada ejercicio.

En la versión inicial del programa no existía ninguna vista donde se pueda observar de un vistazo rápido los resultados de la clase: los ejercicios que ha realizado un alumno, los intentos que hizo y cuántos ejercicios ha conseguido realizar con éxito.

Esta nueva pantalla supone una mejora tanto para el alumno como para el profesor. Para el alumno, aparecer en un listado que incluya a toda la clase le puede motivar a continuar realizando ejercicios para aparecer en mejores puestos de la vista de resultados.

Para el profesor esto supondría una mejora ya que se tendría un control de seguimiento continuo de los alumnos a lo largo del curso. El interés por la realización de ciertos ejercicios deja la puerta abierta a futuras mejoras como por ejemplo la realización de entregas semanales y otras mejoras que aportarían información adicional al profesor para el seguimiento del trabajo de los alumnos.

Ejemplos de uso

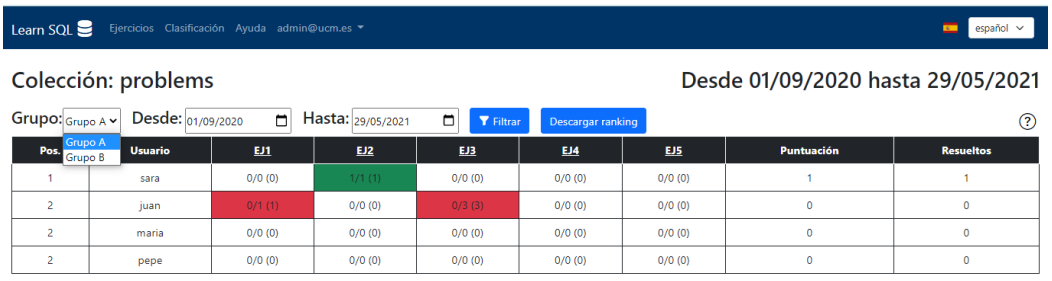
1. **Abrir la vista de colecciones**

   Al pulsar en el botón de "Clasificación" se abre una vista, como se observa en la siguiente imagen, donde poder seleccionar todas las colecciones disponibles

   Al abrir una colección, la vista de la clasificación de la misma cambia según el usuario sea alumno o profesor.

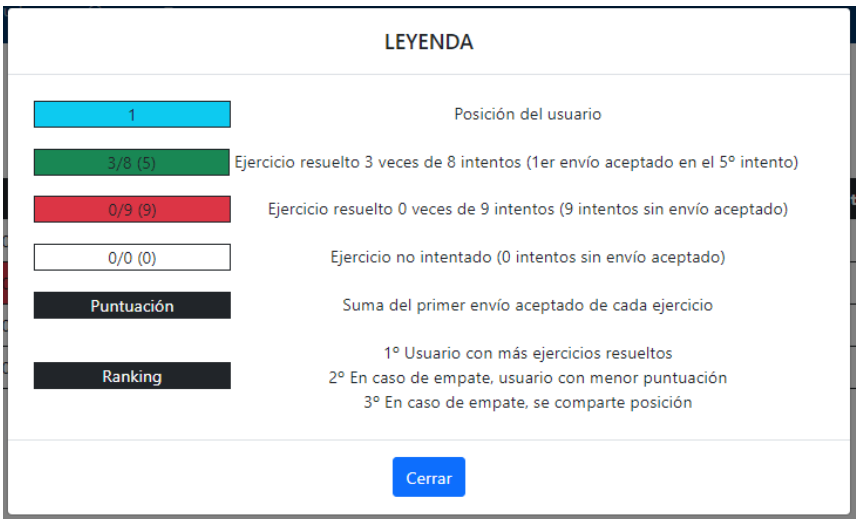
2. **Abrir una colección como profesor**

   Un profesor nunca aparece en la clasificación, al abrir el selector de grupos, como se puede ver en la siguiente imagen, puede elegir todos los grupos existentes.



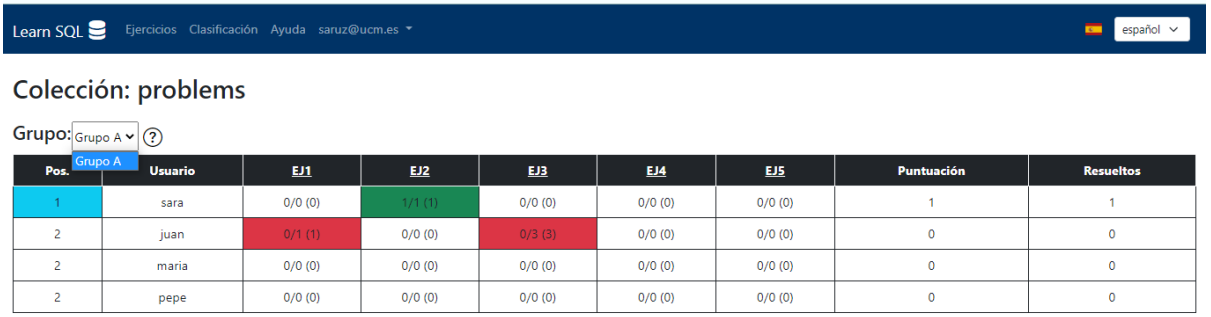
3. **Panel de Información**

   Para entender la clasificación, en la vista existe un círculo con una interrogación en su interior, donde se informará del funcionamiento de la página, como se puede observar a continuación.



4. Abrir una colección como alumno

Al abrirla, se puede ver reflejado en la clasificación gracias a que se nos distingue de los demás compañeros en la posición, la casilla está coloreada de un color azul claro.



Pos.	Usuario	EJ1	EJ2	EJ3	EJ4	EJ5	Puntuación	Resueltos
1	sara	0/0 (0)	1/1 (1)	0/0 (0)	0/0 (0)	0/0 (0)	1	1
2	juan	0/1 (1)	0/0 (0)	0/3 (3)	0/0 (0)	0/0 (0)	0	0
2	maria	0/0 (0)	0/0 (0)	0/0 (0)	0/0 (0)	0/0 (0)	0	0
2	pepe	0/0 (0)	0/0 (0)	0/0 (0)	0/0 (0)	0/0 (0)	0	0

Se puede pinchar en las casillas de los problemas que se han hecho y nos llevará a una vista nueva con nuestros envíos de ese ejercicio en concreto.

5. **Acceder a una clase donde no se tiene acceso**

   En la vista de la clasificación, un alumno puede modificar la URL añadiendo cualquier grupo. En el caso de que eso suceda, saldrá un mensaje de error por pantalla y no tendrán acceso. En la siguiente imagen, se puede observar cómo un alumno del grupo 1, intenta acceder al grupo 2.

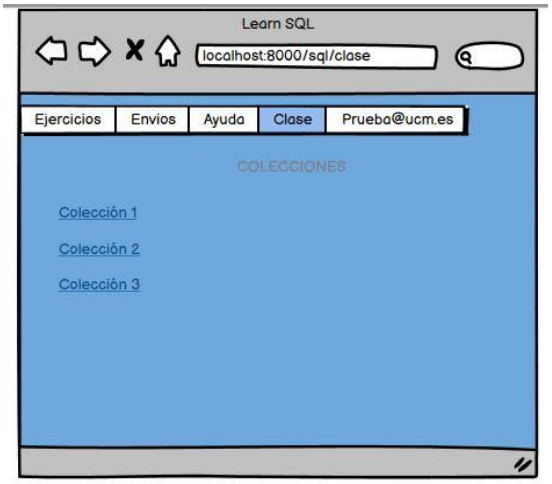
6. **Acceder a los envíos de otro alumno**

   En la vista de los envíos, un alumno puede modificar la URL añadiendo cualquier identificador de usuario. En el caso de que eso suceda, saldrá un mensaje de error por pantalla y no tendrán acceso a los ejercicios del compañero. En la siguiente imagen, se puede observar cómo un alumno con identificador 5, intenta acceder con el identificador 50.

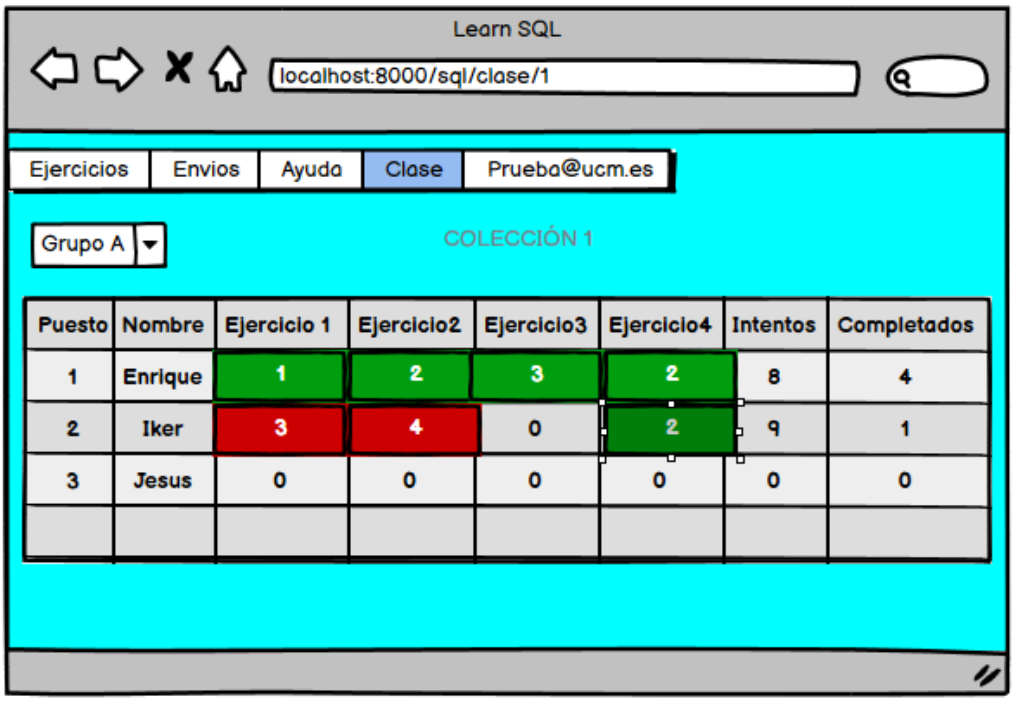
7.4.2 Desarrollo

La idea principal fue obtener una clasificación general de la clase por cada colección existente en el sistema. Al no haber una vista antecesora para su realización, se decidió primero hacer unos *mockups* usando la herramienta Balsamiq [10] como se puede observar en la siguiente imagen.

Se creó un botón en el menú que llevase a una vista genérica que muestre todas las colecciones y al pulsar sobre una colección se muestra la clasificación.



En la vista de la clase se muestra una clasificación donde los alumnos pueden observar los ejercicios que han resuelto, el número de intentos y resueltos (color verde), los ejercicios que han intentado y no hayan resuelto (color rojo) y los ejercicios que no han intentado aún (color blanco) como se puede observar en el *mockup* siguiente.



Un alumno sólo podrá observar la clasificación de su clase. En el selector sólo le aparecerá aquel grupo en el que el alumno esté matriculado. Si el usuario es un profesor, el selector dejará acceder a cualquier grupo.

Puede darse el caso en donde un alumno no esté en ningún grupo. Para estos casos se ha creado una vista informativa en donde se informa al alumno de que no está en ningún grupo.

Desde un punto de vista técnico, la implementación se basa en el uso de parámetros de tipo *GET* [24] que representa al grupo y actualiza la vista según se cambie el selector a otro grupo. Se realiza un control en los parámetros para que un alumno de otro grupo no pueda acceder a un grupo en donde no tiene acceso.

Cada celda contiene un valor que indica el número de intentos realizados hasta el primer *AC* del alumno. Tras varias reuniones, se ha decidido que cada celda debe contener el siguiente formato: *Nº Aciertos / Nº Intentos* (Intento del primer *AC*), donde el valor entre paréntesis indica el primer intento en el que se resolvió con éxito el ejercicio. Este valor es necesario porque puede darse la situación en la que el alumno, después de haber obtenido un resultado correcto, continúe realizando nuevos intentos para mejorar el ejercicio.

Las dos últimas columnas de la vista de resultados contienen información resumida sobre cada alumno. La columna “Intentos” es la suma del primer *AC* en los ejercicios que se han realizado correctamente. En caso de no haberse hecho o que aún no se haya resuelto, no se sumará. La columna “Resueltos” es el número de ejercicios donde al menos haya un envío con veredicto *AC.*

La clasificación en la vista de resultados se organiza conforme al siguiente orden:

1. Se muestran primero los alumnos con un mayor número de ejercicios resueltos.
2. En caso de empate, primero se muestra el usuario con menos Intentos.
3. En caso de empate, ambos alumnos compartirán el mismo puesto en la clasificación.

Para que el método de clasificación sea más entendible, se decidió añadir una leyenda donde se explica el significado de los colores en las celdas y cómo funciona la clasificación.

Durante el transcurso de la tarea se realizaron un total de 102 intercambios de mensajes, repartidos entre: 52 por correo y 42 en el *Pull Request* 26.

7.4.3 Test Unitarios

Las pruebas realizadas para la tarea se basan en la creación de una colección con una serie de ejercicios, dos grupos, dos alumnos y un profesor.

Ambos alumnos pertenecen al primer grupo y el profesor a los dos grupos. La primera prueba consiste en el acceso del profesor a ambos grupos sin problema. Después, un alumno intenta acceder a un grupo inexistente cambiando la URL y para probar la generación de un error "*404 Not Found*".

Ambos alumnos realizan una serie de envíos y se verifica que la clasificación funciona correctamente y se espera que un alumno tenga una mejor posición que el otro alumno.

Por último se realiza una prueba con un alumno nuevo, que no se asigna a ningún grupo y muestra la vista de error indicando que no está asignado a ningún grupo y debe ponerse en contacto con el profesor.

Durante la ejecución de los test se realizan un total de 34 comprobaciones.

7.5 Enlace tabla clasificación envíos

Enlace Pull Request: https://github.com/emartinm/lsql/pull/26

7.5.1 Descripción y motivación

La tarea consiste en poder acceder a los envíos de un ejercicio en concreto desde la vista de resultados desarrollada en la tarea del apartado 7.4. Cada alumno tiene una vista en donde poder ver todos los envíos que ha realizado a lo largo del curso, pero sin filtrar por ejercicios. Esta tarea permite a alumnos y profesores obtener los envíos realizados de un ejercicio en concreto.

Cada alumno podrá pulsar en la fila correspondiente a sus envíos para acceder a la vista de “Mis envíos” del alumno con los envíos correspondientes al ejercicio sobre el que ha pulsado.

La principal motivación de esta tarea se basa en poder acceder rápidamente a los envíos de un ejercicio. De esta forma el profesor puede revisar los envíos de un alumno para un ejercicio sin tener que pasar por todos los envíos que haya hecho el alumno.

7.5.2 Desarrollo

El desarrollo de la tarea se basa en modificar la URL de la vista “Mis envíos” ya que ahora se desean visualizar los envíos que un alumno en particular ha enviado para un ejercicio en concreto. La URL ahora contiene dos parámetros *GET* [24] que son el identificador del problema y el identificador del usuario que ha hecho el problema.

Además debe controlarse el usuario que accede a la página ya que en caso contrario podría acceder cualquier usuario que no fuera ese alumno o un profesor. Un alumno no podrá acceder a los envíos de los demás.

Durante el transcurso de la tarea se realizaron un total de 100 intercambios de mensajes, repartidos entre: 50 por correo y 50 en el *Pull Request* 26.

7.5.3 Test Unitarios

Las pruebas realizadas consisten en comprobar la accesibilidad de los envíos de un alumno: en primer lugar se comprueba que un alumno que ha hecho un envío de un ejercicio en concreto puede acceder a los envíos de ese ejercicio. En segundo lugar se comprueba que un profesor también tiene acceso a los envíos de ese alumno. Por último, se comprueba que otro alumno no puede acceder a la misma url del alumno utilizado en la primera prueba.

Durante la ejecución de los test se realizan un total de 34 comprobaciones.

7.6 Colapsar Tablas

Enlace Pull Request: https://github.com/emartinm/lsql/pull/27

7.6.1 Descripción y motivación

En la versión inicial de la aplicación, todos los ejercicios cuentan con diversas tablas iniciales y la tabla con el resultado esperado y se muestran todas las tablas y sus contenidos en la pantalla. Estas tablas pueden llegar a ser bastante grandes por lo que se vio necesario poder ocultarlas/mostrarlas de manera manual a través de un nuevo botón.

7.6.2 Desarrollo

El desarrollo de esta tarea consiste en añadir a la derecha del nombre de cada tabla un nuevo botón para poder ocultar/mostrar cada una de las tablas que aparecen en los problemas. De esta forma, el contenido de las tablas del problema aparece inicialmente oculto, mostrando solamente los nombres de las columnas, como se muestra en la siguiente figura:

Para poder modificar la plantilla donde se crean las tablas y añadir la nueva función, ha sido necesario buscar documentación sobre Bootstrap [26], ya que ofrece diversas funcionalidades ya predefinidas, entre ellas la función para ocultar/mostrar elementos a partir de un botón. A esto se le sumó la dificultad de poder diferenciar unas tablas de otras a partir de un identificador diferente, por lo que hubo que crear una función aparte para poder asignar un nuevo identificador a cada tabla de un mismo HTML.

Durante el transcurso de la tarea se realizaron un total de 28 intercambios de mensajes, repartidos entre: 19 por correo y 9 en el *Pull Request* 27.

7.6.3 Test Unitarios

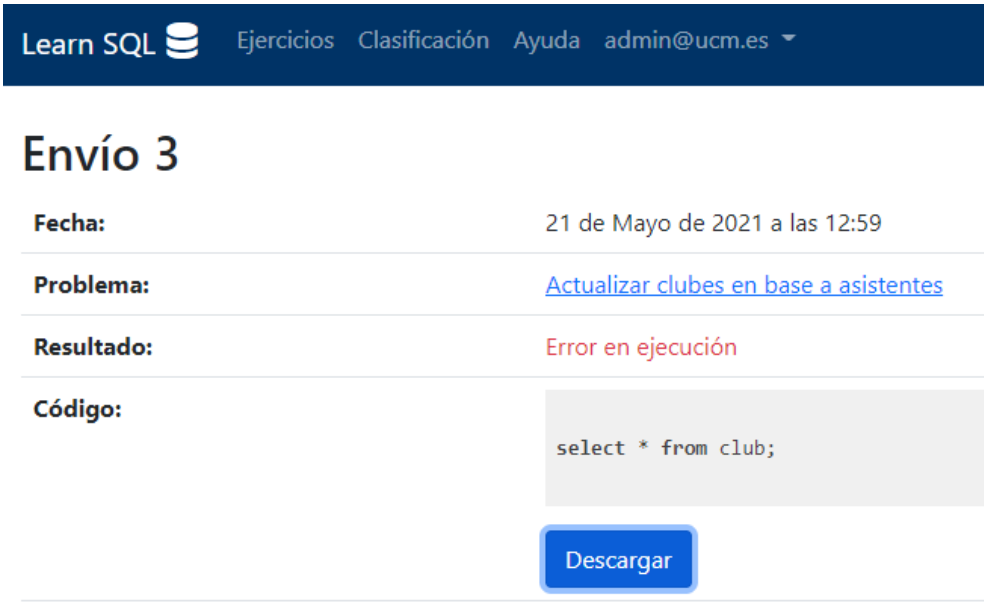
Esta tarea no dispone de pruebas automáticas al ser un código desarrollado íntegramente en HTML, por lo que se decidió no crear ningún test unitario ya que es un funcionamiento visual.

7.7 Descargar Envíos

***Enlace Pull Request*:** https://github.com/emartinm/lsql/pull/28

7.7.1 Descripción y motivación

Esta tarea consiste en un botón para descargar los envíos de los alumnos. En la versión inicial de la aplicación no había ninguna manera de poder descargar un *script* con el código que el alumno envía como respuesta a un ejercicio, solo se podía visualizar en la vista de información de cada envío. Así, con esta tarea se genera un archivo con extensión `SQL` que contiene ese código de envío, como se muestra en la siguiente figura:



La idea ha surgido debido a que los alumnos usan los ordenadores de la facultad para realizar estos ejercicios y para facilitarles el poder descargarse ese código en cualquier ordenador, sin necesidad de tener que llevar consigo un *pendrive* o enviar el código por correo. De esta manera nunca perderán esa información.

7.7.2 Desarrollo

Esta tarea fue la primera tarea seleccionada por el componente del equipo que se encargó de ella. Para poder implementarla hubo que investigar sobre Django ya que era una herramienta nueva y desconocida.

Tras haber entendido la estructura del proyecto y comprendido el funcionamiento de Django con el tutorial oficial [1], el desarrollo ha consistido en la creación de un botón en cada página de envío para cada problema. Una vez el botón es pulsado se realiza una llamada a una función que crea un fichero `SQL` con la información del código enviado por el alumno que se descarga en el ordenador. Las principales dificultades que han aparecido en esta tarea han estado relacionadas con la realización de los test de unidad, ya que era novedad al no haber trabajado antes con pruebas unitarias. Otra complicación fue averiguar que había que decodificar el contenido que devolvía el campo *response.content* [24], puesto que estaba en *bytes* y se necesitaba tener una cadena de caracteres.

Durante el transcurso de la tarea se realizaron un total de 18 intercambios de mensajes en el *Pull Request* 28.

7.7.3 Test Unitarios

Las pruebas realizadas para esta tarea se han basado en el funcionamiento del botón. Se ha comprobado que al realizar la descarga del archivo, éste contiene exactamente el mismo código del envío y que la propiedad de la cabecera corresponde a un archivo `SQL`. También se ha verificado que un alumno, registrado o no, no puede acceder a esa URL y descargar el código de otro alumno. Y por último, que el profesor sí tuviera acceso al código de todos los alumnos.

Durante la ejecución de los test se realizan un total de 7 comprobaciones.

7.8 Campo para enviar un archivo como respuesta

***Enlace Pull Request*:** https://github.com/emartinm/lsql/pull/33

7.8.1 Descripción y motivación

Antes de la finalización de esta tarea, para realizar los envíos de las soluciones de los problemas, los alumnos debían escribir en el campo de texto que se encuentra en la parte inferior o copiando y pegando desde una herramienta externa en la que pudiera haber creado el código, pudiendo copiar algunos caracteres no visibles que hicieran que el resultado no fuera el esperado.

Se pensó en crear un botón para ayudar al alumno a entregar el ejercicio, pudiendo crear el código en un editor aparte y después simplemente arrastrar el archivo al juez para comprobar que funciona correctamente.

De esta forma, la tarea consiste en crear un campo con un botón en el que se pueda agregar un archivo, ya sea arrastrando en el espacio disponible o buscando en los archivos locales del ordenador con el botón “Explorar”. Una vez el archivo esté seleccionado correctamente se debe ver el nombre del archivo para que el alumno esté seguro de que ha seleccionado el correcto y también debe haberse rellenado el campo donde debe ir el código con el que está escrito en el archivo. En la imagen siguiente se puede observar el botón creado:

7.8.2 Desarrollo

Para poder desarrollar esta tarea hubo que familiarizarse con el funcionamiento de Ace Editor [26], el editor de texto donde los alumnos introducirán su código, ya que al

cargar el archivo con el botón que se iba a crear se debía introducir el código del archivo en ese editor de texto.

Lo primero que se realizó fue la creación del botón tipo archivo, el cual nos permite arrastrar archivos o seleccionarlos de nuestro propio ordenador con el botón. Una vez se selecciona el archivo se llama a una función Javascript donde se lanza todo el proceso para que el contenido del archivo pase al editor de texto. Se tuvo que crear un *script* para que al subir el archivo se mostrara el nombre del archivo en lugar de “Elige un archivo”.

Una de las mayores dificultades encontradas durante el desarrollo de esta tarea fue justamente el poner el nombre del archivo, ya que al llamar al *script* se introducía en el HTML el nombre del archivo local, produciendo una vulnerabilidad XSS (*Cross-site scripting*) en la que se ejecutaría el código insertado en el nombre del archivo local, pudiendo producir graves problemas de seguridad. Se evitó este riesgo cambiando la llamada al *script* con un atributo distinto al anterior, el cual ya no inserta directamente en el HTML el nombre del archivo.

Otro de los problemas encontrados fue que solo dejaba subir archivos con el formato `TXT`, esto se debía a la restricción incluida en el propio formato del botón. Se solucionó quitando la restricción en la creación del botón.

Durante el transcurso de la tarea se realizaron un total de 16 intercambios de mensajes en el *Pull Request* 33.

7.8.3 Test Unitarios

Al ser un código desarrollado íntegramente en Javascript y HTML se decidió no crear ningún test unitario ya que este botón tiene un funcionamiento exclusivamente visual.

7.9 Filtro de fechas en la vista de resultados

Enlace Pull Request: https://github.com/emartinm/lsql/pull/42

Enlace Issue: https://github.com/emartinm/lsql/issues/34

7.9.1 Descripción y motivación

Esta tarea consiste en añadir dos filtros de fechas, exclusivo para los profesores, con los que se puede filtrar la clasificación (apartado 7.4). De esta forma, los profesores pueden tener un mayor control de los periodos en los que los alumnos han realizado los ejercicios, si han sido constantes durante todo el curso, o bien han realizado los ejercicios antes de empezar los exámenes.

Inicialmente la fecha de inicio es el 1 de septiembre del curso académico actual y la fecha final es el día de hoy. El profesor puede modificar las fechas y aplicar de nuevo el filtro que desee.

7.9.2 Desarrollo

Se hizo un *mockup* de cómo quedaría la clasificación con los filtros de las fechas para una primera aprobación del resultado.

Como se puede observar en la imagen anterior, para tener en cuenta los filtros, a la URL se han añadido dos parámetros “Start” y “End” con el formato “*ddmmyyyy*”.

Este filtro es exclusivo para los profesores, los alumnos no verán los calendarios ni el botón de filtro. De la misma forma, también se controla el acceso a la vista utilizando una URL con las fechas del filtro, pues en caso de que sea un alumno, no se debe permitir entrar a la vista.

Al ajustar la vista con las fechas, también se debe modificar la vista de los envíos realizados por los alumnos si se pulsa sobre alguna de las casillas de la vista de resultados. De este modo, la vista "Mis Envíos" también se ha visto modificada para que, aparte de filtrar por identificador del problema y el usuario, también se filtre entre dos fechas concretas.

Se valida que el formato de fechas sea el correcto y que la fecha inicio sea menor o igual que la fecha final. Se ha realizado un formulario proporcionado por Django para el control de las fechas y tener controlado en un mismo sitio todos los posibles errores que se puedan producir [27].

Durante el transcurso de la tarea se realizaron un total de 62 intercambios de mensajes en el *Pull Request* 42.

7.9.3 Test Unitarios

Las pruebas realizadas para la tarea consisten en añadir en las pruebas realizadas de la clasificación casos para modificar las fechas de los envíos de los alumnos. En particular, se añaden dos fechas de inicio correctas y se comprueba que entre esos rangos sólo aparecen los ejercicios que han sido enviados dentro del rango.

También se comprueba que el formato de fechas sea el correcto, que la fecha final sea mayor o igual que la fecha inicio y que los parámetros de fecha no se encuentren vacíos.

Durante la ejecución de los test se realizan un total de 29 comprobaciones.

7.10 Podio

Enlace Pull Request: https://github.com/emartinm/lsql/pull/43
Enlace Issue: https://github.com/emartinm/lsql/issues/36

7.10.1 Descripción y motivación

Esta tarea fue pensada y desarrollada con la finalidad de que los alumnos, de manera rápida y sencilla, pudiesen saber quiénes han sido los tres primeros en resolver cada ejercicio desde el listado de problemas, sin la necesidad de tener que ir a la clasificación, y así intentar animar y conseguir que resuelvan los ejercicios para poder estar en la nueva interfaz de “Top 3” de cada ejercicio implementada en esta tarea.

7.10.2 Desarrollo

El primer paso para este desarrollo consistió en crear una función que, dado un problema y una posición, devolviese al usuario que ha resuelto el problema en esa posición [28], teniendo en cuenta solamente el primer envío correcto de cada usuario. Tras esto, se diseñó el estilo que tendría en la interfaz de la lista de problemas. Se plantearon varias alternativas de visualización a medida que se iba desarrollando la tarea. Primero se añadieron tres nuevas columnas con los títulos “Primero”, “Segundo” y “Tercero”. Este diseño no terminaba de agradar y se decidió buscar unos iconos [29] para resaltar que la función de dichas columnas es premiar a los tres primeros usuarios, por lo que se añadieron unas medallas de color “Oro”, “Plata” y “Bronce”. Finalmente se decidió cambiar los títulos anteriores por “1º”, “2º” y “3º” ya que así quedaría más compacto y agradable a la vista.

La interfaz quedó de la siguiente manera:

Colección 1

Descripción de la primera coleccion.

Listado de problemas

Nombre	Número de envíos	1º	2º	3º
Actualizar clubes en base a asistentes	2	Pepe	-	-
Extraer partes de una cadena	2	Andrés	Juan	Pepe
Modificar las filas insertadas/modificadas	1	Pepe	-	-
Filtrar columnas 4	50	Juan	Pepe	-
Insertar una fila con clave externa	2	-	-	-

Durante el transcurso de la tarea se realizaron un total de 41 intercambios de mensajes, repartidos entre: 23 en el *Pull Request* 43 y 18 en el *Issue* 36.

7.10.3 Test Unitarios

Los tests unitarios de esta nueva función consistieron en comprobar que la función creada para devolver el usuario que ha resuelto cierto problema en una determinada posición solamente devolviese usuarios cuyos envíos fuesen correctos y comprobar que devolviese a los usuarios en el orden correcto sin tener en cuenta ningún envío más además del primer envío correcto de cada usuario.

Durante la ejecución de los test se realizan un total de 21 comprobaciones.

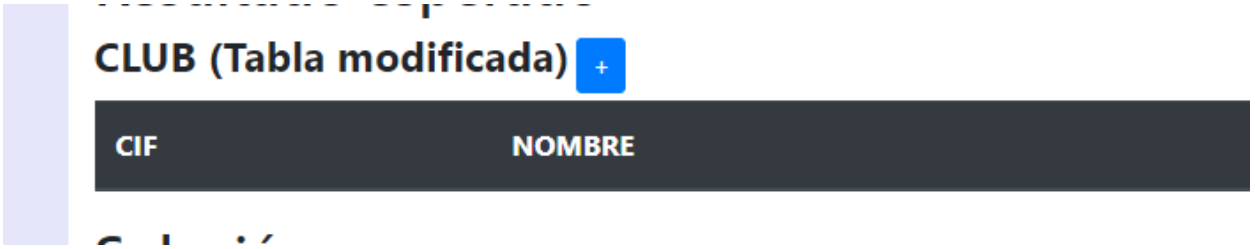
7.11 Mostrar tablas modificadas

Enlace Pull Request:
https://github.com/emartinm/lsql/pull/47
https://github.com/emartinm/lsql/pull/44
Enlace Issue: https://github.com/emartinm/lsql/issues/37

7.11.1 Descripción y motivación

Se debía mejorar el resultado esperado en los problemas de tipo DML, disparador y procedimiento, ya que este tipo de problemas espera que se cambie la base de datos inicial, se pueden eliminar, añadir o modificar varias tablas. Antes de esta mejora se

mostraba como resultado esperado todas las tablas de la base de datos, tanto las modificadas como las que no se veían afectadas por la solución del problema. Tras esta tarea los alumnos van a poder encontrar más fácilmente los cambios solicitados en el enunciado del problema. En la siguiente figura se puede observar el texto al lado del nombre de la tabla.



La tarea consiste en mostrar, al lado de cada tabla del ejercicio, el cambio que se espera que se produzca como resultado del código DML, disparador o procedimiento. Los mensajes que pueden salir al lado del nombre de la tabla son:

- Tabla modificada si alguno de los valores de la tabla inicial se ha cambiado, si se ha añadido alguna fila o columna y si se ha eliminado alguna fila o columna.
- Tabla añadida si la tabla no existía en la base de datos inicial.
- Tabla eliminada si la tabla ahora no debería existir pero sí aparecía en las tablas iniciales.

7.11.2 Desarrollo

Al inicio de la tarea se pensó en hacer esta mejora solo para los problemas de procedimientos, finalmente se llegó a la conclusión que se podía reutilizar el trabajo realizado para mejorar la salida de las tablas en los problemas DML y los de tipo disparador.

Se creó una función en la que se filtran las tablas y se obtienen las tablas que se eliminan o se añaden y las tablas que no se modifican. Las tablas que no cumplan ninguno de los requisitos anteriores son las tablas modificadas.

Gracias a la potencia de Django para trabajar con plantillas se crearon 3 plantillas nuevas, heredando de la plantilla principal de los problemas, para mostrar el resultado esperado una vez filtradas las tablas.

Durante el transcurso de la tarea se realizaron un total de 82 intercambios de mensajes, repartidos entre: 55 en los *Pull Request* y 27 en el *Issue* 37.

7.11.3 Test Unitarios

Los test unitarios para comprobar el correcto funcionamiento de esta tarea comprueban la nueva función creada y que las tablas se muestran correctamente en la vista. Se crean dos funciones de test nuevas. La primera de ellas se utiliza para comprobar el funcionamiento de la función mencionada en el apartado anterior, se crean dos bases de datos, la inicial y la esperada, en la que se elimina una tabla, se añade otra y se modifica una de las iniciales, tras esto se compara con la base de datos esperada.

La segunda función de test se utiliza para comprobar que, una vez que se ejecuta la función anterior y llegamos a la vista del problema, se muestran en la cabecera de la tabla los mensajes correspondientes: “Tabla añadida”, “Tabla modificada” o “Tabla eliminada”.

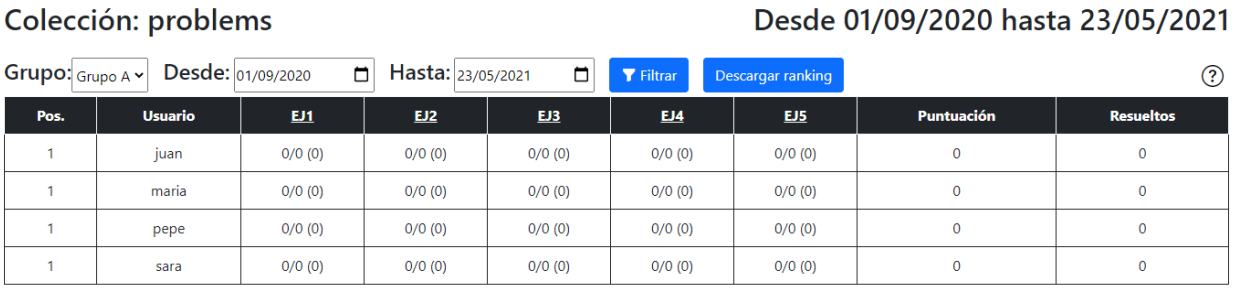
Durante la ejecución de los test se realizan un total de 8 comprobaciones.

7.12 Descargar ranking

Enlace Pull Request: https://github.com/emartinm/lsql/pull/48
Enlace Issue: https://github.com/emartinm/lsql/issues/35

7.12.1 Descripción y motivación

Esta tarea consiste en la creación de un botón para descargar la clasificación de los alumnos. Así los profesores podrán descargarse la información de la clasificación de sus alumnos con el botón que se muestra en la siguiente figura:



Colección: problems Desde 01/09/2020 hasta 23/05/2021

Grupo: Grupo A Desde: 01/09/2020 Hasta: 23/05/2021 Filtrar Descargar ranking

Pos.	Usuario	EJ1	EJ2	EJ3	EJ4	EJ5	Puntuación	Resueltos
1	juan	0/0 (0)	0/0 (0)	0/0 (0)	0/0 (0)	0/0 (0)	0	0
1	maria	0/0 (0)	0/0 (0)	0/0 (0)	0/0 (0)	0/0 (0)	0	0
1	pepe	0/0 (0)	0/0 (0)	0/0 (0)	0/0 (0)	0/0 (0)	0	0
1	sara	0/0 (0)	0/0 (0)	0/0 (0)	0/0 (0)	0/0 (0)	0	0

La idea ha surgido a raíz de la creación de una vista de resultados (ver apartado 7.4). No había ninguna manera de poder descargar la información de la clasificación general de la clase para cada colección y grupo existente en el sistema. Así se facilita esta información al profesor y este podrá descargarse toda la clasificación de la tabla en un archivo en formato `XLS` y tener un mejor manejo de dicha información.

7.12.2 Desarrollo

Para poder desarrollar esta tarea hubo que investigar qué biblioteca era la más adecuada para poder manejar toda la información en formato `HTML` y escribirla en un archivo de extensión `XLS`. Se realizaron varias pruebas con distintas bibliotecas de python para escribir archivos `XLS` sin éxito, hasta encontrar la biblioteca *openpyxl* [30]. El desarrollo ha consistido en la creación de un botón, solo disponible para el profesor. Una vez pulsado el botón se realiza una llamada a una función que reutiliza la página web generada en el apartado 7.4 y se analiza el HTML para extraer la información y generar un archivo `XLS` que contiene dicha información de la tabla de clasificación. Las principales dificultades encontradas en esta tarea han sido el manejo de distintas bibliotecas de Python. Se han usado las siguientes bibliotecas: *BeautifulSoup* [31], *openpyxl* [30], *tempfile* [32]; la primera biblioteca se ha utilizado para analizar el código HTML de la página de vistas (apartado 7.4). La segunda biblioteca se ha usado para crear un *workbook* donde se almacenaba la información de la tabla en *worksheets.* La tercera biblioteca se ha usado para guardar ese *workbook* en un fichero temporal con nombre.

Durante el transcurso de la tarea se realizaron un total de 76 intercambios de mensajes, repartidos entre: 37 en el *Issue* 35 y 39 en el *Pull Request* 48.

7.12.3 Test Unitarios

Las pruebas realizadas para esta tarea se han basado en el uso del botón. Se ha comprobado que al realizar la descarga contuviera exactamente la misma información incluida en la tabla de clasificación y que la cabecera correspondiera a un archivo de extensión `XLS`. También se ha verificado que al modificar la *URL* con datos de fecha o grupos inexistentes o erróneos se devuelven los errores correspondientes a cada situación. Además, se ha comprobado que un alumno no pueda acceder a esa *URL* y descargar el ranking de clasificación. Y por último, que el profesor sí pueda descargarse el ranking.

Durante la ejecución de los test se realizan un total de 14 comprobaciones.

7.13 Logros

Enlaces Pull Request:

https://github.com/emartinm/lsql/pull/49

https://github.com/emartinm/lsql/pull/56

https://github.com/emartinm/lsql/pull/70

***Enlaces Issues*:**

https://github.com/emartinm/lsql/issues/39

https://github.com/emartinm/lsql/issues/54

https://github.com/emartinm/lsql/issues/67

7.13.1 Descripción y motivación

Esta tarea consiste en desarrollar un sistema de logros que pueden conseguir los alumnos realizando distintas misiones mientras se solucionan problemas de las colecciones de problemas disponibles en la aplicación. Con este sistema de logros se pretende motivar a los alumnos a hacer ejercicios y hacer una especie de competición entre ellos y consigo mismos para que así sigan resolviendo ejercicios.

Los logros debían ser configurables por los profesores para agilizar el trabajo en el momento de crearlos. También se debían actualizar los conseguidos cuando se crea

uno para poder añadir los logros a lo largo del curso para seguir motivando a los alumnos a seguir consiguiendolos.

Los tres tipos de logros creados han sido los siguientes:

- *NumSolvedAchievement*: Realizar un envío correcto en un número determinado de problemas distintos. Por ejemplo “Resuelve 5 problemas”, siendo el 5 el número determinado por el profesor.
- *PodiumAchievement*: Resolver un número de problemas entre los primeros alumnos. Por ejemplo “Resuelve 3 problemas entre los 5 primeros”, donde 3 y 5 son los valores parametrizables por el profesor al definir el logro.
- *NumSolvedCollectionAchievement*: Enviar una solución correcta a un número determinado de problemas de una misma colección. Por ejemplo “Resuelve 2 problemas de la colección Navidad 2021”, en este caso “Navidad 2021” es el nombre de la colección que el profesor puede configurar. El número de problemas a resolver también es un valor configurable.

Tras realizar el desarrollo de otras tareas se decidió incluir dos tipos nuevos de logros para seguir dando motivación a los alumnos en distintos apartados, también configurables por el profesor, además de poder añadirse a lo largo del curso. Los logros que se añadieron fueron los siguientes:

- *NumSolvedTypeAchievement*: Solucionar un número determinado de problemas de un mismo tipo. Por ejemplo “Resuelve 3 ejercicios de SELECT”, donde 3 y “SELECT” serían los valores modificables por el profesor.
- *NumSubmissionsProblemsAchievement*: Realizar envíos, tanto correctos como incorrectos a un cierto número de problemas. Este logro se creó para seguir motivando al alumno a realizar envíos para intentar resolver el ejercicio aunque realizara muchos envíos fallidos. Por ejemplo “Haz 4 envíos en un total de 3 problemas”, donde 4 y 3 son los valores parametrizables por el profesor.

Ejemplos de uso

Se va a crear un logro de cada tipo, se iniciará sesión con un usuario previamente creado y se van a conseguir cada uno de los distintos logros. El sistema tendrá más de un usuario para poder comprobar la vista del ranking y la vista de logros de un usuario distinto al propio, además tendrá varios problemas en distintas colecciones para poder comprobar el funcionamiento de los distintos logros.

1. **Crear los logros**

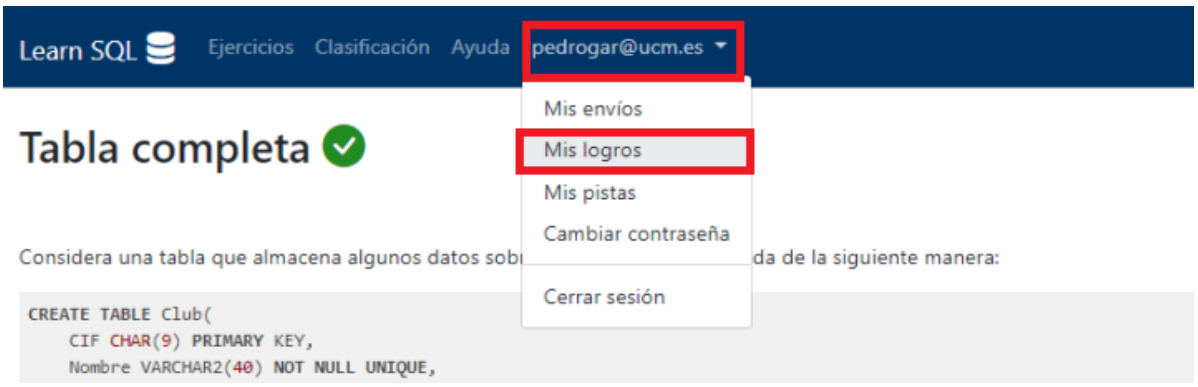
   Primero, en la página de administrador se crean los distintos logros. El primer logro creado es del tipo *NumSolvedAchievement,* se configura de tal forma que para conseguirlo se necesite solamente acertar un problema. También se crea un logro *PodiumAchievement* en el que haya que resolver un problema entre los 3 primeros alumnos. El logro *NumSolvedCollectionAchievement* se configura para que el alumno tenga que acertar un problema de la colección “TFG 2021”. El logro *NumSolvedTypeAchievement* se configura para que haya que acertar un problema del tipo “SELECT”. Por último, se crea un logro *NumSubmissionsProblemsAchievement* configurado para que haya que realizar dos envíos a cualquier problema para poder obtenerlo.

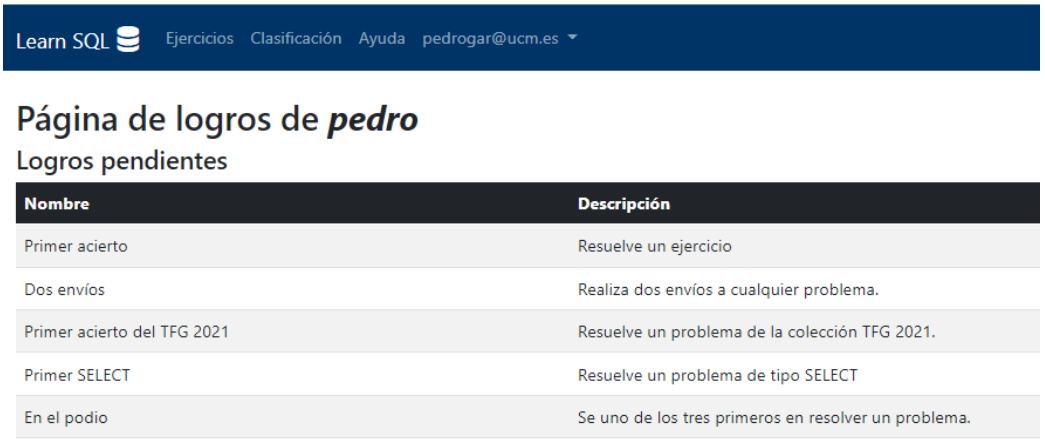
   En la siguiente figura se puede observar la vista del administrador donde se pueden ver todos los logros creados.

2. Abrir la vista de logros

Una vez configurados los logros por parte del profesor, se inicia sesión con una cuenta de alumno para realizar los envíos. Primero se entra en la pestaña de "Logros" para ver qué logros están disponibles, se puede ver que están todos como "pendientes", ya que no se ha empezado a resolver problemas.

En las figuras que se muestran a continuación se puede ver cómo se llega a la vista de "Mis logros" y la vista de los logros del usuario con el que se ha iniciado sesión.

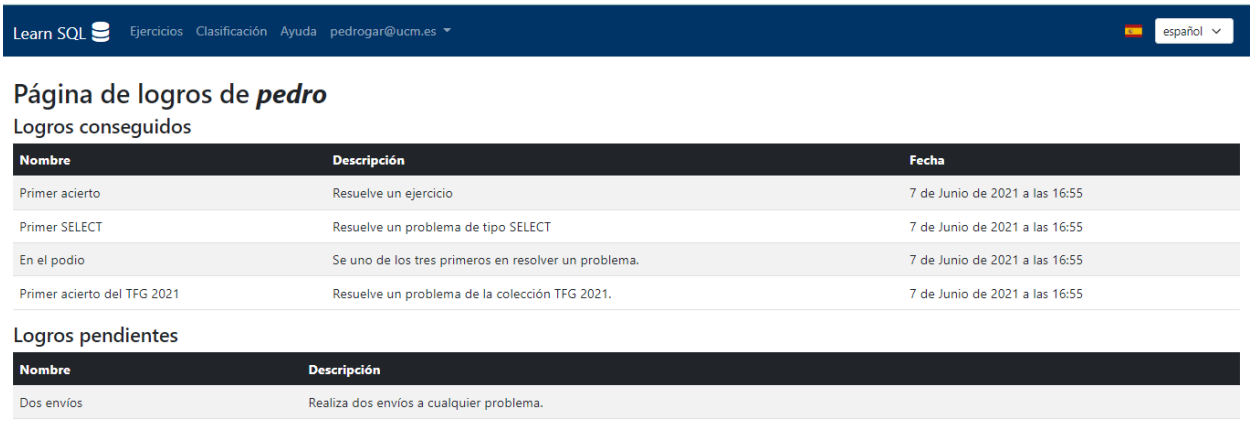




3. Realizar un envío para conseguir los logros

Se va a realizar un envío correcto en un problema de tipo "SELECT" de la colección "TFG 2021". Con este envío se han conseguido todos los logros excepto *NumSubmissionsProblemsAchievement.* Se va a volver a la página de

"Logros" para poder observar cómo se muestran los logros. Ahora se puede ver un logro como pendiente y el resto como conseguidos.

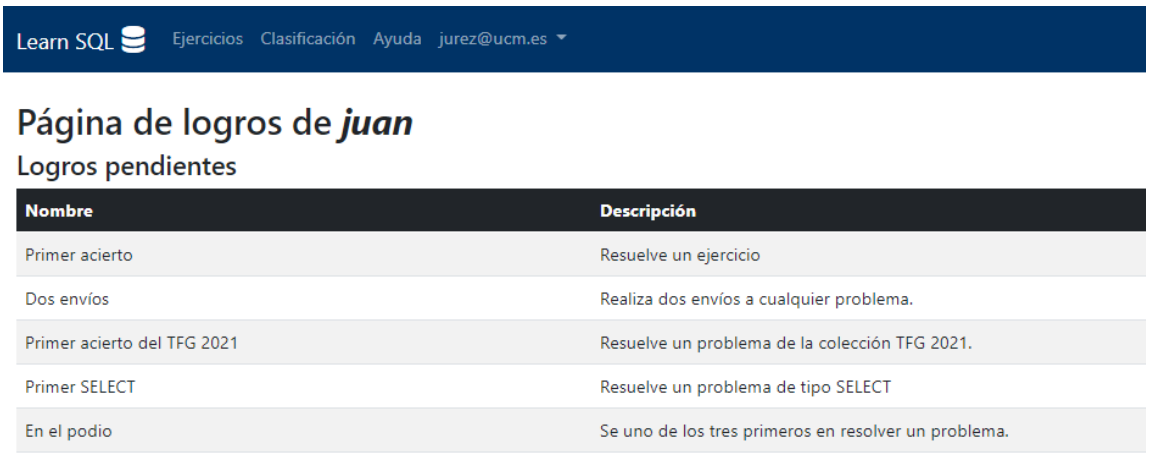


4. Entrar en el ranking

Ahora se puede acceder a la vista del ranking, donde se puede observar que el único usuario que tiene el trofeo es el usuario con el que se ha realizado el envío, el resto de usuarios no ha conseguido ningún logro hasta el momento.

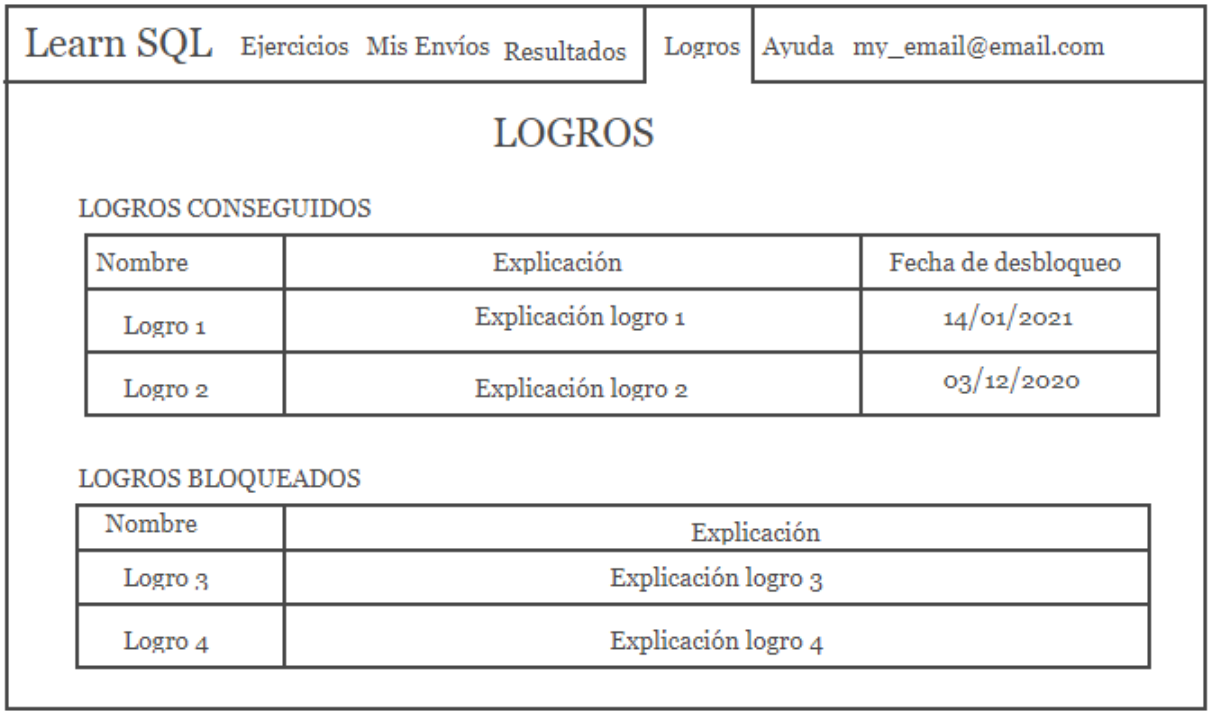
5. Entrar en la página de logros de otro usuario

Se pulsa sobre el nombre de alguno de los usuarios para entrar a su pestaña de logros y así se puede observar que están todos como "pendientes".



7.13.2 Desarrollo

Una de las partes más importantes de esta tarea era crear la vista para que los alumnos pudieran visualizar qué logros hay disponibles para desbloquear y cuáles tienen ya conseguidos. Se creó un *mockup* para que los integrantes del grupo y los tutores pudieran valorar el resultado que se iba a buscar conseguir:



Se realizaron algunas matizaciones para la vista. La pestaña logros estaría incluida en un menú desplegable junto al correo. También los nombres de las tablas serían diferentes y se irían discutiendo a lo largo de la creación de la vista puesto que era un detalle secundario al inicio. Una vez aprobado este diseño se empezó con la implementación.

Lo primero que se implementaron fueron los modelos de Django [33]. Para simplificar el código se creó un modelo que nos serviría como interfaz o clase abstracta, de la que heredarían los modelos de los logros, ya que cada logro iba a tener un modelo diferente. Así se conseguían unificar algunas funciones que iban a ser compartidas por los logros, como la comprobación de que un usuario determinado tuviera el logro que se estaba comprobando. También se creó un modelo donde se almacenarían los logros conseguidos por cada usuario, almacenando la fecha de obtención del logro, el logro y el usuario que lo ha conseguido.

Para almacenar los logros obtenidos había que conseguir la fecha exacta en la que el usuario consiguió realizar este logro, esto fue un reto ya que los profesores tienen la posibilidad de crear logros nuevos a lo largo del curso y se debe comprobar si los usuarios, con los envíos ya realizados, han conseguido este logro nuevo. Por tanto el logro podía crearse el 10 de mayo de 2021 pero el usuario había conseguido los requisitos para desbloquearlo el 3 de mayo de ese mismo año, así pues debía almacenarse la fecha del 3 de mayo para que el usuario sepa en qué fecha exacta consiguió este logro. Para poder realizar esto se tuvo que investigar en el campo de las señales de Django [34], las cuales sirven para hacer comprobaciones antes o después de realizar una llamada a la base de datos, de forma similar a un disparador. En este caso se creó una señal que, cada vez que se hiciera la llamada a la base de datos creando un logro, se comprobaran los usuarios que lo habían conseguido y poniendo la fecha de ese último envío como fecha de obtención del logro.

Tras realizar la parte de los modelos y las señales se pasó a crear la nueva vista que finalmente quedó como se muestra a continuación.

LOGROS

Logros conseguidos

Nombre	Descripcion	Fecha
Empiezas bien	Resuelve un problema	March 22, 2021, 6:48 p.m.
Segundo	Sé el segundo o el primero en resolver un problema	March 22, 2021, 6:48 p.m.

Logros pendientes

Nombre	Descripcion
Primero	Sé el primero en algún problema
Resolvista	Resuelve 3 problemas
Rey de los Tigres	Resuelve tres problemas de la colección Triggers

Además se propuso que cualquier alumno pudiera ver los logros de sus compañeros para así fomentar la competición con otros alumnos Para esto se agregó en la tabla de la clasificación un trofeo con un número al lado, con el número de logros conseguidos por cada uno de los usuarios.

Colección: Coleccion de Triggers

Grupo: Grupo Pruebas

Pos.	Usuario	EJ1	EJ2	EJ3	EJ4	EJ5	Puntuación	Resueltos
1	willyrex x5	1/1 (1)	1/2 (2)	0/1 (1)	1/1 (1)	0/0 (0)	4	3
2	michu x2	0/0 (0)	0/0 (0)	0/0 (0)	1/1 (1)	0/0 (0)	1	1
3	immobile	0/0 (0)	0/0 (0)	0/0 (0)	0/0 (0)	0/0 (0)	0	0

Al pulsar en los trofeos se accede a la página de logros del usuario enlazado para ver los logros que ha conseguido este usuario. Con esta mejora se tuvo que crear un nuevo *Pull Request* para mejorar la vista de logros y así quedara visible el nombre del usuario del cual estamos viendo sus logros.

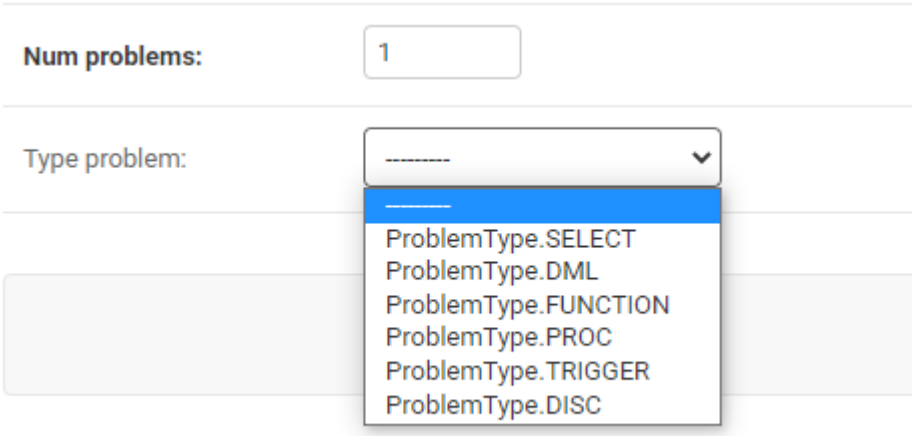
Página de logros de ***daniel***

Logros pendientes

Nombre	Descripcion
Resuelve 1	1

Antes de realizar los dos últimos tipos de logros descritos en el apartado anterior se realizaron otras tareas, para ir añadiendo otras funcionalidades. Se esperó a terminarlas para poder ampliar el abanico de logros disponibles que puede crear el profesor.

Con las vistas creadas el mayor reto encontrado con estos nuevos logros fue la implementación de los tipos de problemas, ya que se podría dar el caso de que en un futuro se crearan nuevos tipos. Por este motivo, la lista de tipos debía actualizarse automáticamente en el modelo del logro. Se cogió la lista de tipos de la base de datos para crear un selector en la configuración del logro, así esta lista se iría actualizando cada vez que se agregara o eliminara un tipo de problema.



Durante el transcurso de la tarea se realizaron un total de 181 intercambios de mensajes, repartidos entre: 125 en los *Pull Request* y 56 en los *Issues*.

7.13.3 Test Unitarios

Se decidió crear una clase nueva para los test de esta tarea ya que eran muy extensos y si se incluían en la clase de test creada con anterioridad se sobrepasarían las 1000 líneas de código, que es el tamaño máximo permitido para mantener un código legible por la herramienta Pylint.

Se crearon 9 funciones de test para probar las distintas implementaciones realizadas, como se detalla a continuación. Tres test para comprobar que se creaban correctamente cada uno de los logros y además se podían desbloquear con los requisitos creados en el test; un test para comprobar la vista del ranking, para confirmar que el número al lado del trofeo indica correctamente el número de logros conseguidos

por el usuario; dos test para comprobar las señales, uno de ellos para verificar que al crear un logro se lanza la señal y otro para cerciorarse de que al actualizar uno de los logros también se envían estas señales; los otros test comprueban que las fechas de obtención de los logros son correctas y que los errores lanzados por las funciones se muestren correctamente.

Se añadieron los nuevos logros en estas funciones ya creadas con anterioridad para verificar su correcto funcionamiento.

Durante la ejecución de los test se realizan un total de 16 comprobaciones.

7.14 Bases de datos iniciales

Enlace Pull Request: https://github.com/emartinm/lsql/pull/52
***Enlace Issue*:** https://github.com/emartinm/lsql/issues/38

7.14.1 Descripción y motivación

Esta tarea consiste en la posibilidad de tener varias bases de datos iniciales diferentes para cada ejercicio de tipo “SELECT”. La principal motivación para llevar a cabo esta tarea fue que, en determinadas ocasiones, para un problema se puede proponer una solución que funciona correctamente con la base de datos del enunciado, pero no funciona con otros datos de prueba.

Esta tarea tiene como objetivo conseguir ser más concretos con la solución a la que hay que llegar para poder resolver el ejercicio de la manera que el profesor propone. Por este motivo se necesitaba poder tener más de una base de datos inicial, para evitar estas situaciones y que la solución anterior, que es correcta para la primera base de datos, no lo sea para la segunda, y así poder concretar más la solución esperada. En el enunciado solo se mostraría la base de datos principal del ejercicio.

Ejemplos de uso

Se van a enviar varias soluciones a un problema que tiene varias bases de datos iniciales para mostrar la respuesta que daría el sistema al enviar una solución correcta

para todas las bases de datos iniciales, para alguna de ellas pero no todas y para ninguna de ellas.

Las tablas de la base de datos inicial del problema son las siguientes:

Base de datos Descargar script

CLUB +

CIF	NOMBRE	SEDE	NUM_SOCIOS
11111111X	Madrid	A	70000

PERSONA +

NIF	NOMBRE
00000001X	Peter Johnoson

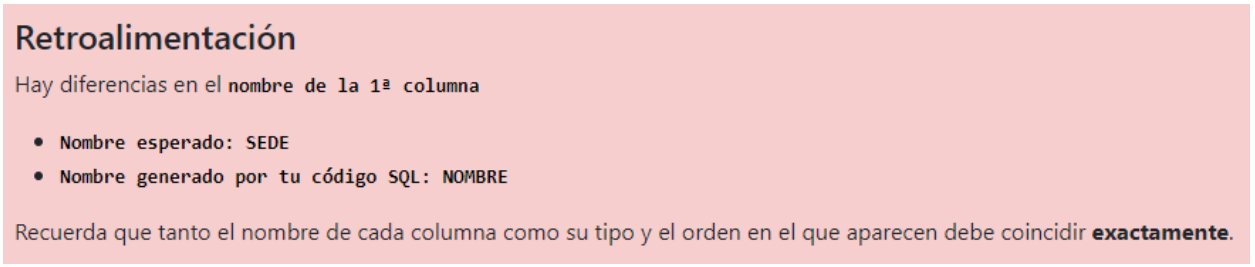
Resultado esperado

SEDE	NOMBRE
A	Madrid

En el enunciado se pide que en la solución se muestre exclusivamente el club con CIF igual a “11111111X” y de nombre “Madrid”.

1. **Fallo en la base de datos del enunciado**

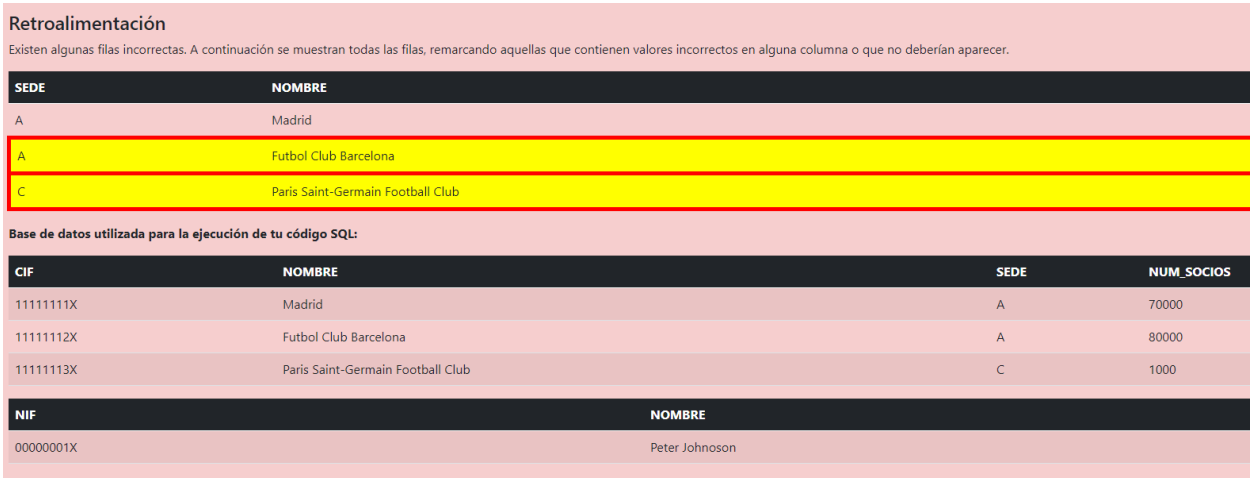
   En caso de enviar una solución que no muestre el resultado esperado, como por ejemplo "SELECT NOMBRE, SEDE FROM CLUB", se obtendrá el siguiente mensaje indicando que el esquema obtenido no es el esperado:



2. **Fallo en una base de datos secundaria**

   Si se envía una solución que devuelve el resultado esperado, pero no de la manera indicada (CIF igual a “11111111X” y nombre “Madrid”), como por ejemplo, "SELECT SEDE, NOMBRE FROM CLUB", ya que en este caso solo hay un valor y mostraría el que se ve en el resultado esperado, esta solución sería válida si solamente se contase con la base de datos inicial del enunciado, pero

al haber incluido más bases de datos iniciales en el problema gracias a esta tarea, se podrá mostrar el siguiente mensaje de error:



Retroalimentación

Existen algunas filas incorrectas. A continuación se muestran todas las filas, remarcando aquellas que contienen valores incorrectos en alguna columna o que no deberían aparecer.

SEDE	NOMBRE
A	Madrid
A	Futbol Club Barcelona
C	Paris Saint-Germain Football Club

Base de datos utilizada para la ejecución de tu código SQL:

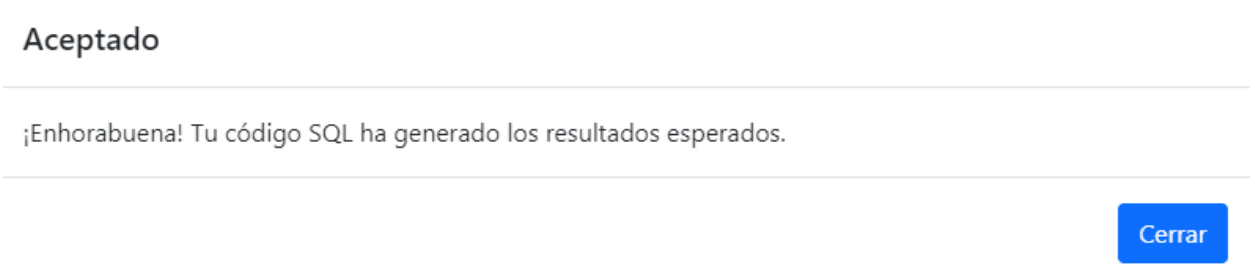
CIF	NOMBRE	SEDE	NUM_SOCIOS
11111111X	Madrid	A	70000
11111112X	Futbol Club Barcelona	A	80000
11111113X	Paris Saint-Germain Football Club	C	1000

NIF	NOMBRE
00000001X	Peter Johnoson

Se muestra el resultado que se obtiene al hacer la consulta en la primera base de datos inicial que no sea la del enunciado y en la que se produzca un resultado incorrecto, mostrando en amarillo, en este caso, las filas de más que se obtienen, y a continuacion las tablas utilizadas para obtener dicho resultado y así mejorar la respuesta que obtiene el alumno, mostrándole en qué caso ha fallado, ya que esta base de datos inicial no aparece en el enunciado.

3. **Solución válida en todas las bases de datos**

Finalmente, si el alumno envía una solución válida para todas las bases de datos iniciales, el sistema notificará que se ha enviado la respuesta correcta:



7.14.2 Desarrollo

El primer paso que hubo que dar para comenzar con este desarrollo fue cambiar la manera en que se almacena la base de datos inicial de cada ejercicio, ya que en la versión inicial del sistema cada problema contaba con una sola base de datos. Lo que se propone con esta tarea es que un problema pueda tener varias bases de datos, por lo que el campo que guardaba dicho elemento se cambió a una lista de elementos. Como este desarrollo sólo tenía en cuenta los problemas de tipo “SELECT”, para el resto de problemas también se cambió a una lista pero solo se utilizaba el primer elemento en todos los lugares del código donde se necesitase. Una vez realizado este cambio, la siguiente cuestión consistia en decidir cómo agregar más de una base de datos inicial desde la interfaz de administrador.

La base de datos inicial de un problema se almacena como dos cadenas de caracteres. Una cadena de instrucciones DDL para crear las tablas y otra cadena de instrucciones para insertar las filas. Se decidió modificar la segunda para representar múltiples bases de datos iniciales, en la que cada conjunto de instrucciones para rellenarlas esté separado entre sí por una cadena que funciona como separador. Mediante una función se detectan los distintos fragmentos de sentencias de inserción y así guardarlas en la lista anteriormente mencionada. Tras esto, el siguiente paso fue cambiar el código de tal manera que no solo se comprobase la solución con la primera base de datos, sino que se probase con todas en orden, de tal manera que el ejercicio no devolviese un resultado correcto hasta que se comprobase la solución con todas las bases de datos.

Finalmente, se añadió la función de mostrar las tablas de la primera base de datos con la que la solución proporcionada no devolviese resultado correcto, a excepción de la primera, ya que esta se muestra por defecto en el enunciado del problema. Para esto hubo que rehacer la estructura de las plantillas de solución incorrecta, creando una nueva plantilla general para todas donde se muestre la base de datos, en caso de que se necesite mostrar, de la cual heredasen el resto de plantillas de solución incorrecta.

Durante el transcurso de la tarea se realizaron un total de 43 intercambios de mensajes, repartidos entre: 15 en el *Pull Request* 52 y 28 en el *Issue* 38.

7.14.3 Test Unitarios

Para el funcionamiento de las múltiples bases de datos iniciales se crearon tests para comprobar que un ejercicio solo aparece como correcto si, y sólo si, la solución proporcionada es correcta para todas las bases de datos y también se crearon tests para comprobar que en la retroalimentación de solución incorrecta, aparece la base de datos en la que ha fallado la solución y que no muestre ninguna en caso de fallar en la base de datos principal.

Durante la ejecución de los test se realizan un total de 20 comprobaciones.

7.15 Ejercicios de discriminación de consultas

Enlace Pull Request:
https://github.com/emartinm/lsql/pull/63
https://github.com/emartinm/lsql/pull/64
Enlace Issue: https://github.com/emartinm/lsql/issues/51

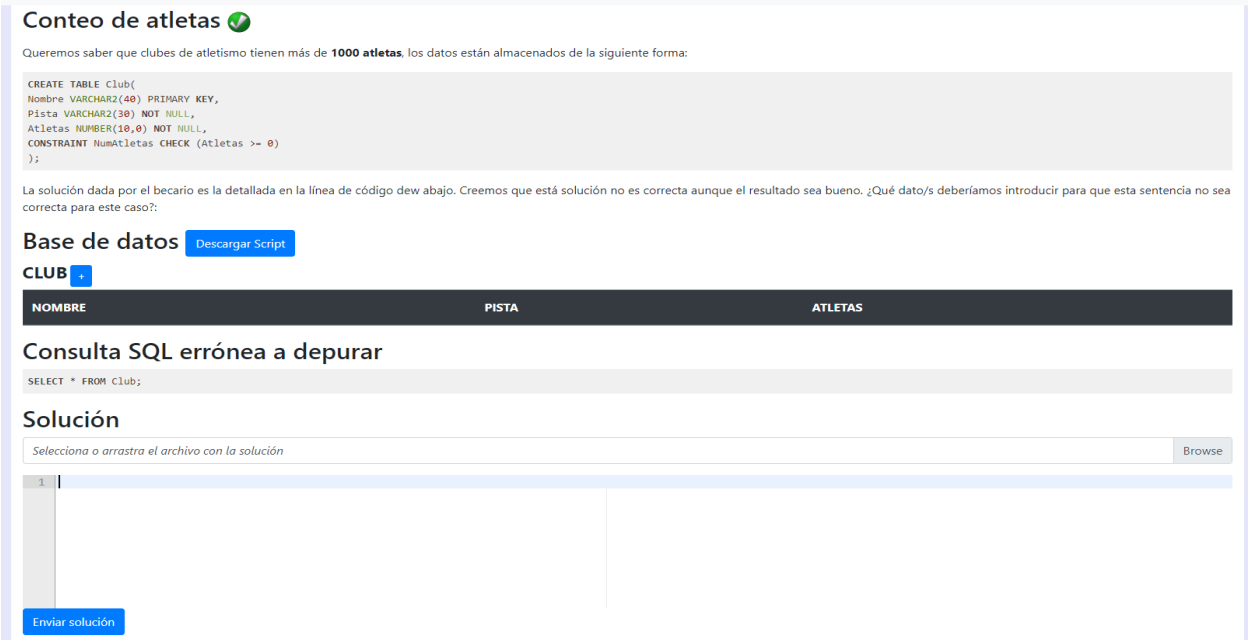
7.15.1 Descripción y motivación

Se crea un nuevo tipo de problema no existente en la versión inicial de Learn SQL para que los alumnos puedan aprender sobre la diversidad de respuestas que tiene un mismo problema y, aunque el resultado sea correcto para la base de datos de pruebas, la solución no siempre es correcta para todas las bases de datos.

El enunciado del problema está formado por los siguientes elementos: una base de datos, un enunciado y una solución que produce un resultado correcto sobre la base de datos, pero que no es exactamente lo que se pide ya que esa solución con otra base de datos producirá un resultado incorrecto. Se pide al alumno que proporcione una o varias sentencias que modifiquen los datos de la base de datos de forma que la respuesta dada por la solución del enunciado sea incorrecta.

Estos problemas, al igual que los demás tipos de problemas, deben de poderse introducir en la base de datos como un archivo `ZIP` para así agilizar el proceso de creación de problemas para el profesor y que se puedan introducir varios problemas a

la vez. En la siguiente figura se puede observar la página del problema, en la que se ve el título, el enunciado y el campo para subir la solución.



7.15.2 Desarrollo

En primer lugar se estudiaron los distintos problemas ya creados en la plataforma para así adecuar lo máximo posible el formato a los ya existentes. Se creó un modelo nuevo para el problema, heredando del modelo *Problem.* Se crearon los atributos que almacenan tanto la consulta incorrecta mostrada al alumno como la consulta correcta que estará oculta pero nos servirá para comprobar la solución del alumno. Se implementaron las funciones necesarias para tratar este nuevo tipo de problemas.

La función más relevante que se implementó fue la función *judge,* que compara los resultados de las bases de datos para comprobar si el resultado dado por el alumno es correcto. Lo que se hace para comprobar la solución del alumno es ejecutar las sentencias de modificación de datos proporcionadas por el alumno en las tablas de la base de datos inicial, y a continuación se ejecutan las consultas del problema, tanto la correcta como la incorrecta. Después de estas ejecuciones quedan dos resultados: si

estos son iguales la solución dada por el alumno no será correcta, ya que no se habrá detectado el fallo de la consulta incorrecta.

Se ajustaron los mensajes de error para que el alumno tuviera más claro dónde podía haber cometido un fallo, si es que así hubiera sido.

Se creó una vista nueva con una plantilla nueva extendida de la vista problemas para que en el enunciado se mostrara la consulta incorrecta que el alumno debe depurar.

Durante el transcurso de la tarea se realizaron un total de 111 intercambios de mensajes, repartidos entre: 50 en el *Issue* 51 y 61 en los *Pull Request*.

7.15.3 Test Unitarios

Se crearon varios test para comprobar el correcto funcionamiento de este nuevo problema. Se crearon dos funciones distintas, una que comprueba el mensaje de error devuelto con cada uno de los tipos de respuesta que se pueden producir (*AC*, *WA* o *RE*) y la otra función que comprueba que tras enviar una solución el resultado dado es el correcto. Se comprobó que la inserción del problema fuera correcto al crearse. Además de todas las comprobaciones pertinentes del `ZIP`, para esto se reutilizo código de las comprobaciones de otros problemas.

Durante la ejecución de los test se realizan un total de 9 comprobaciones.

7.16 Soporte multidioma

Enlaces Pull Request:
https://github.com/emartinm/lsql/pull/68
https://github.com/emartinm/lsql/pull/74
https://github.com/emartinm/lsql/pull/76
https://github.com/emartinm/lsql/pull/77
***Enlace Issue*:** https://github.com/emartinm/lsql/issues/62

7.16.1 Descripción y motivación

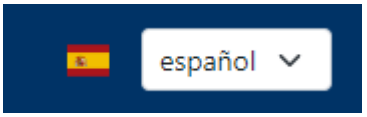
La idea de esta tarea surgió a raíz de poder extender el uso de Learn SQL en varios idiomas simultáneamente. El ámbito más cercano al cual se le va a dar uso será para alumnos extranjeros que estén estudiando en la universidad, ya sea por erasmus o por otros motivos, y para el grupo en inglés del Grado en Ingeniería Informática. Por tanto, esta tarea se basa en adaptar la aplicación de tal manera que sea fácil y sencillo incorporar nuevos idiomas para su uso y poder cambiarlo en cualquier momento a través de un menú desplegable en la interfaz de usuario.

7.16.2 Desarrollo

El primer paso y el más importante fue incorporar la posibilidad de soportar varios idiomas, para ello se utilizó como referencia la documentación de Django sobre la internacionalización [35].

Una vez hecho esto se añadieron dos idiomas, el español, utilizado por defecto, y el inglés. A continuación hubo que cambiar en el código todas las referencias a frases y palabras que aparecían en la interfaz por “enlaces” a los archivos de las traducciones donde se encontraban todas las frases y palabras traducidas al idioma determinado por el archivo. De esta forma, en función del idioma en que se encontrase la página, dichos “enlaces” se sustituirán por el contenido de cada documento de traducción. Esta primera función y parte de la traducción de la web se añadieron con el primer *Pull Request*.

El segundo paso fue añadir un botón con un menú desplegable para el cambio de idioma a través de la interfaz de usuario.



El tercer paso fue añadir iconos de banderas [36] en función del país a las colecciones de ejercicios, para poder diferenciar los distintos idiomas de los problemas sin necesidad de tener que entrar en ellos para averiguarlo, de tal manera que, a la

derecha del nombre de cada colección, apareciesen las banderas de los idiomas de los problemas contenidos en cada colección, a excepción del caso en el que todos los problemas estuviesen en español, ya que de momento es el idioma principal y el más usado, para no sobrecargar la interfaz con iconos. Aun así, esta excepción se puede cambiar, ya que está creado como un filtro para poderse modificar. Al añadir la biblioteca con las diferentes banderas, se pensó que también sería buena idea poner la bandera del idioma actual al lado del selector de idiomas.

Por último, se añadió la posibilidad de que los logros contasen con múltiples idiomas para así poder mostrarlos en el idioma correspondiente en función del idioma actual seleccionado en la interfaz, y la traducción de la página de ayuda, de tal manera que en función del idioma seleccionado, se cargue una plantilla u otra con el idioma correspondiente.

Durante el transcurso de la tarea se realizaron un total de 165 intercambios de mensajes, repartidos entre: 90 en los *Pull Request* y 75 en el *Issue* 62.

7.16.3 Test Unitarios

Las funcionalidades comprobadas a partir de los tests son el correcto funcionamiento de la selección de idioma, es decir, que el selector de idioma funciona correctamente y la página se muestra en el idioma seleccionado, tanto para texto como para los logros, la página de ayuda, la retroalimentación de los ejercicios, etc, además de la adecuada aparición de las banderas de idioma en función de los problemas contenidos en cada

una. Para esta tarea se decidió crear una nueva clase específica para los tests relacionados con los idiomas.

Durante la ejecución de los test se realizan un total de 136 comprobaciones.

7.17 Pistas

Enlaces Pull Request:
https://github.com/emartinm/lsql/pull/78
https://github.com/emartinm/lsql/pull/84
https://github.com/emartinm/lsql/pull/85
Enlace Issue: https://github.com/emartinm/lsql/issues/40

7.17.1 Descripción y motivación

En la versión inicial de la aplicación no existía ningún sistema de pistas que sirviera de ayuda a los alumnos cuando tienen dificultades para resolver los ejercicios. Gracias a esta extensión, cada ejercicio puede incluir un número de pistas que se desbloquean cuando el alumno realiza el número de envíos configurados en la pista. En caso de no haber pistas para un determinado ejercicio, el botón que activa una ventana emergente donde se muestran las pistas aparece deshabilitado. Este sistema también cuenta con un aviso para el alumno que le informa en todo momento de las consecuencias de pedir una pista, pues al pedir una pista se podrá ver afectada la clasificación del alumno en el ranking. En este aviso también aparecerá información adicional sobre las pistas: cuántas pistas hay disponibles para ese ejercicio, cuando se queda sin pistas, a cuántos envíos está de desbloquear la siguiente pista o si todavía no ha solicitado ninguna pista. Para solicitar pista, habrá un botón que solo estará disponible si quedan pistas para ese ejercicio, si no permanecerá desactivado. Al mismo tiempo, el sistema dispone de una vista donde se recopila la información de todas las pistas usadas por cada alumno organizada por problema y ordenado por la fecha en la que las usó. Finalmente, para facilitar la incorporación de más pistas para un problema, el profesor podrá cargar las pistas desde un fichero *markdown* almacenado junto a los demás archivos de los `ZIP` de los problemas.

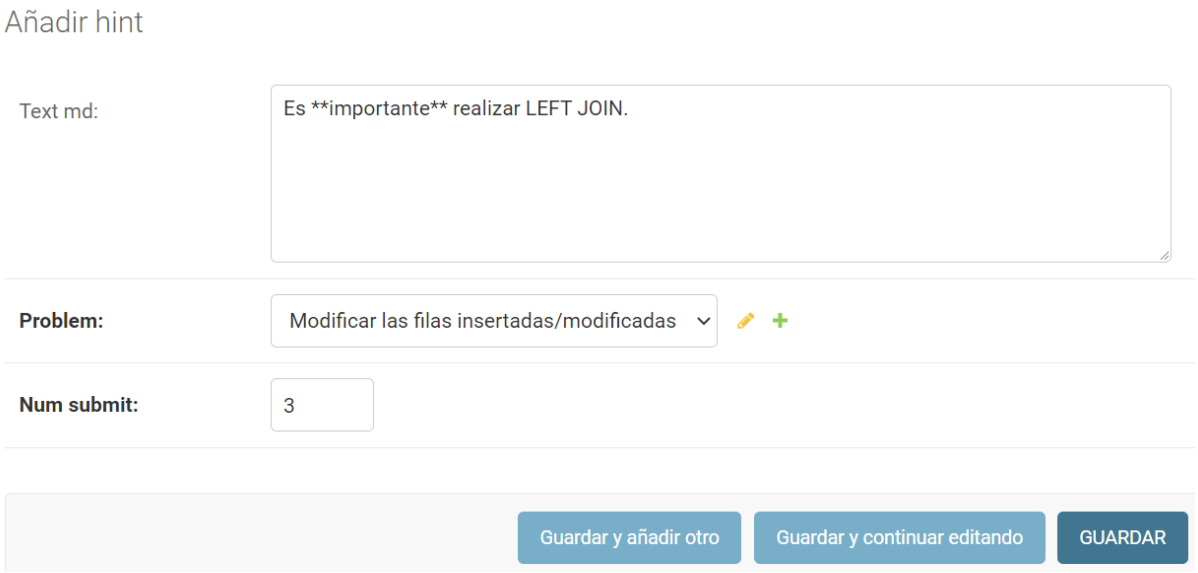
La idea ha surgido debido al abandono o poca participación de los alumnos en algunas asignaturas de este tipo donde se usan jueces en línea. Con este sistema se pretende motivar al alumnado y dar un mayor apoyo a los alumnos que se encuentren más perdidos en la asignatura.

Ejemplos de uso

Se va ha acceder a la plataforma iniciando sesión como alumno y se procederá a pedir pistas para un ejercicio con pistas disponibles y para otro que no tiene ninguna pista disponible. También se accederá a la vista de “Mis pistas” donde estarán a nuestra disposición todas las pistas usadas para cada uno de los ejercicios.

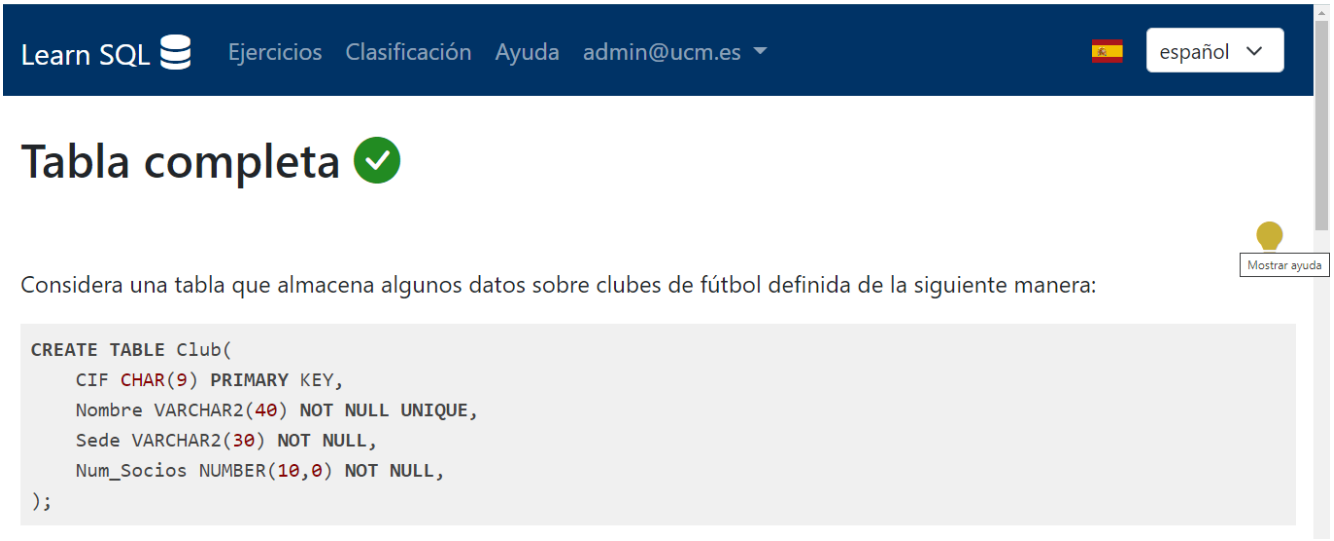
1. **Crear pistas para un ejercicio**

   La siguiente imagen se corresponde con la creación de una pista para un problema concreto, en este caso "Modificar las filas insertadas/modificadas" y que va a requerir de tres envíos para poder proporcionar esta pista.

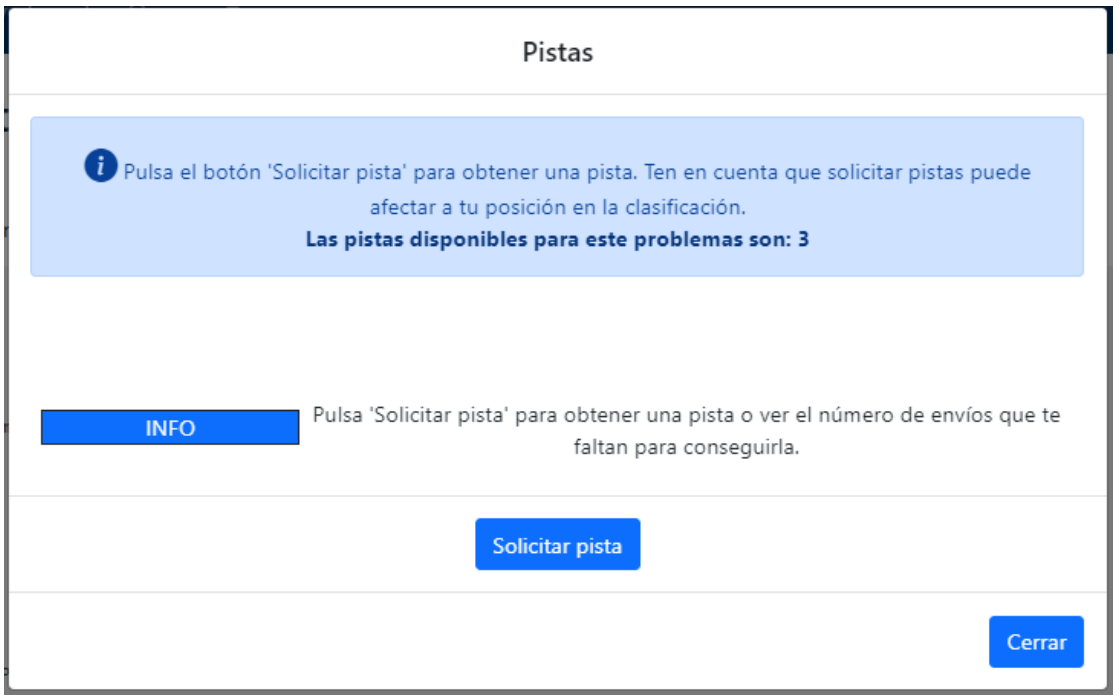


2. Solicitar pistas para un ejercicio

En la vista de un problema concreto, en este caso el de "Modificar las filas insertadas/modificadas". En la parte derecha de la pantalla debajo del título del problema aparece un botón con una bombilla, como se puede ver en la siguiente imagen:

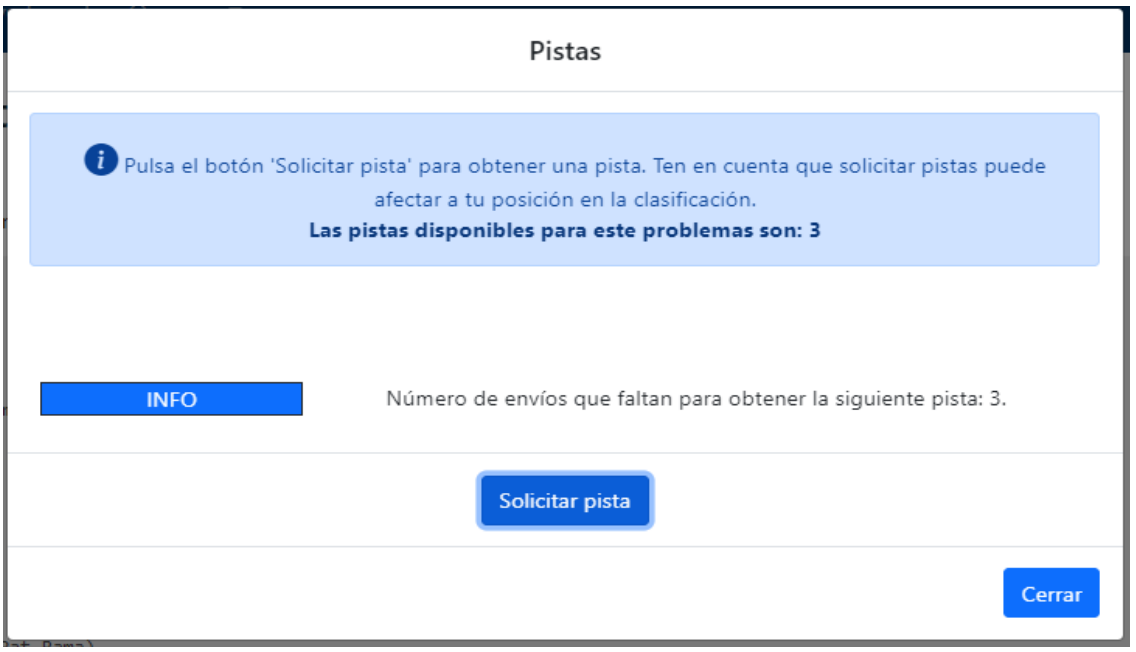


En este caso, como el problema tiene pistas disponibles, aparecerá de color amarillo. Pulsamos el botón y se mostrará una vista como la que se indica a continuación:



Como se puede ver en la imagen anterior, en el recuadro de información siempre se le comunicará al alumno el número de pistas disponibles en el ejercicio y que solicitar una pista puede afectar su puntuación en el ranking.

Se procederá a la solicitud de una pista. En el caso de no tener los envíos necesarios para poder solicitarla, se mostrará un mensaje como el siguiente indicando cuántos envíos faltan para que esté disponible esa petición:

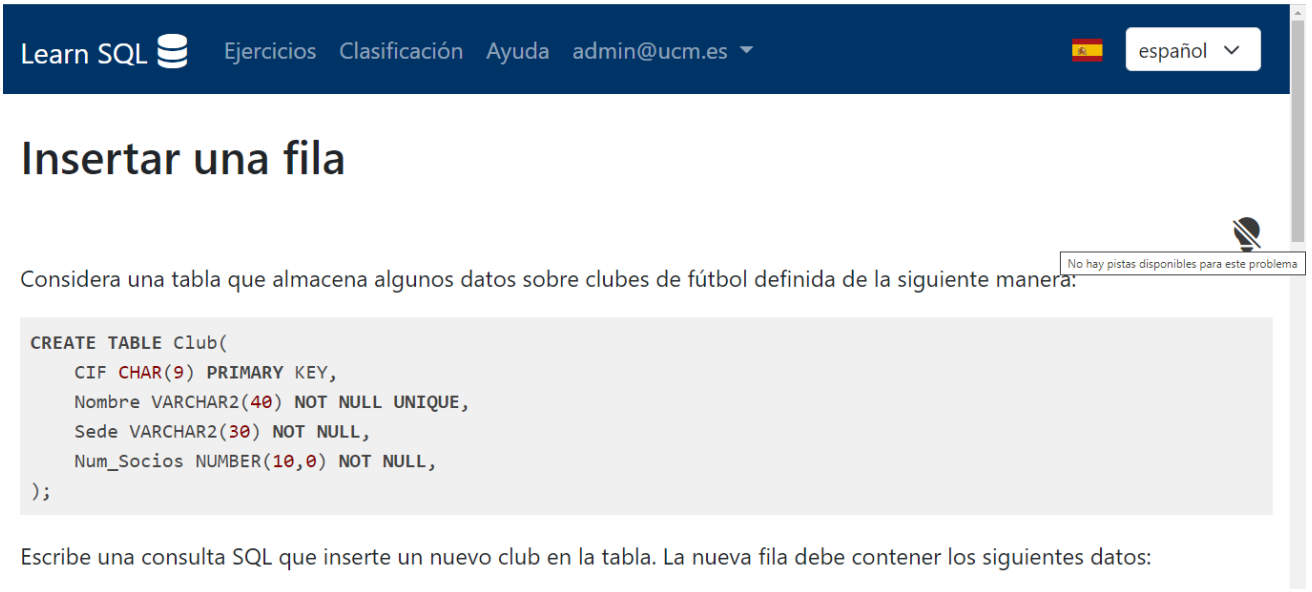


Después de realizar los envíos necesarios indicados anteriormente, ya tendríamos disponible la pista. Al solicitarla se mostrará la pista como se puede observar en la siguiente figura:

Tras haber solicitado todas las pistas y haber agotado las pistas disponibles para este problema, se verá un mensaje informativo indicando que ya no quedan más

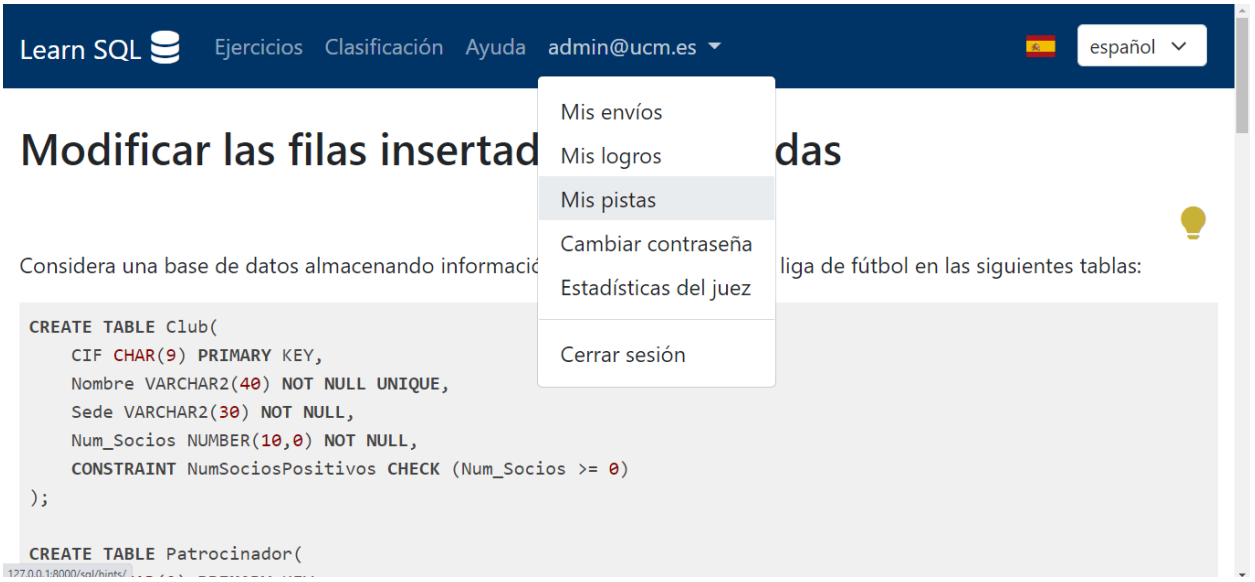
pistas disponibles para ese ejercicio y se desactivará el botón "Solicitar pista", tal y como se muestra a continuación:

Por otro lado, si estamos en un ejercicio sin pistas disponibles, el botón de la bombilla aparecerá desactivado y de color gris, indicando que esa funcionalidad no esta disponible como se puede ver en la siguiente imagen:



3. Visualizar la vista de tus pistas usadas

Para poder ver todas tus pistas usadas ordenadas por fechas y agrupadas por problema, pulsamos en "Mis pistas" como se ve a continuación:

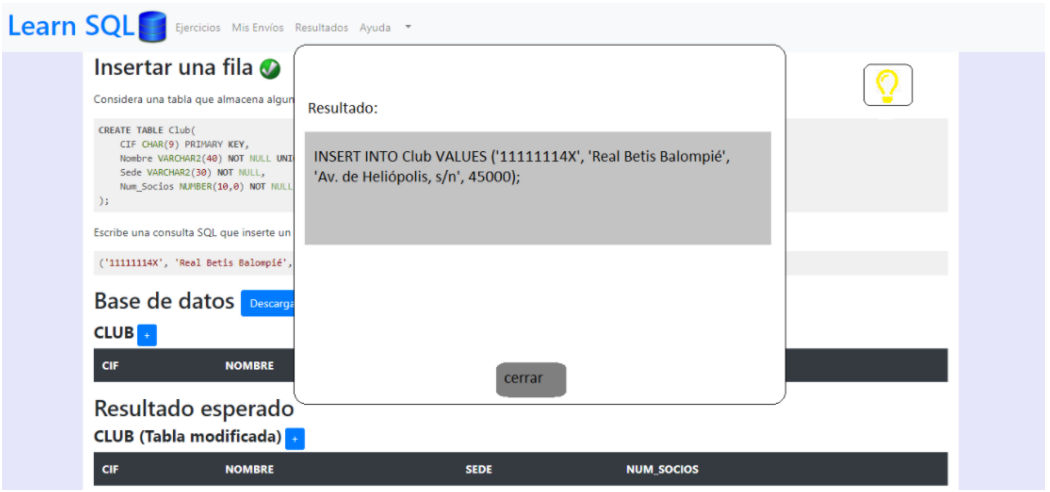


Esta acción nos dirigirá a una vista con toda la información de las pistas usadas por el usuario tal y como se muestra en la siguiente imagen:

Como se observa, el título de cada problema te redirecciona al problema correspondiente.

7.17.2 Desarrollo

Para poder desarrollar esta tarea se hizo un *mockup* que sirviera de base para que se fuera modificando y desarrollando la idea.



Hubo que crear dos modelos nuevos [33]: uno para las pistas disponibles de un ejercicio y otro para las pistas usadas por los alumnos. El modelo de las pistas incluye la descripción de la pista, una clave foránea que apunta al problema al que pertenecía y el número de envíos necesarios para obtener esa pista. El modelo de pistas usadas contiene la fecha en la que se solicitó la pista y dos claves foráneas, una que apunta al

alumno que ha solicitado la pista y otra que apunta al objeto 'pista'. Para la visualización de las pistas se diseñó una ventana emergente que contuviera toda información sobre las pistas y un botón que al ser pulsado realiza una llamada asíncrona a una función que contesta a esta petición *POST* con un JSON que modifica el estado del HTML de la ventana emergente.

En una primera versión se diseñó una ventana emergente con cinco zonas: aviso, ayuda, lista de pistas, info y los botones de solicitar pista y cerrar la ventana. La primera zona de aviso se ha dedicado para informar al alumno que solicitar una pista tiene un impacto en la clasificación del ranking, la segunda zona de ayuda se ha destinado a informar sobre el número de pistas disponibles que tiene el problema, la tercera zona se ha designado a mostrar la información de las pistas, la cuarta zona está dedicada a indicar al alumno si puede pedir o no una pista nueva ya sea por no estar disponible por falta de envíos en ese ejercicio o por no haber más pistas disponibles.

En la versión final del diseño se unificaron las dos primeras zonas anteriores en una única a modo de mensaje de alerta.

Para esta tarea además hubo que implementar otra vista dedicada a mostrar la información de las pistas usadas por un alumno en tablas diferenciadas para cada ejercicio. Por último, se implementó una función para poder leer las pistas desde un archivo *markdown* almacenado dentro de la carpeta comprimida en `ZIP` que contiene el problema. De esta forma se pueden cargar las pistas de manera automática en caso de que existan pistas para ese problema, sin necesidad de añadir las pistas una a una a mano. Para poder guardar las pistas desde el archivo *markdown* hubo que usar señales [34] para poder crear correctamente las pistas de un problema concreto justo después de haberlo salvado en la base de datos.

Las principales dificultades que han aparecido en esta tarea fueron las siguientes: la complejidad de la función de Javascript dedicada a establecer el contenido de la ventana emergente a partir del `JSON` recibido, la adición de atributos dinámicos y la comprobación de los mensajes de error de las excepciones lanzadas al leer un archivo de pistas. Otra complicación fue el manejo de señales para poder almacenar las pistas de un problema ya que las pistas dependen de la existencia de un problema.

Durante el transcurso de la tarea se realizaron un total de 334 intercambios de mensajes, repartidos entre: 78 en el *Issue* y 256 entre los tres *Pull Request*.

7.17.3 Test Unitarios

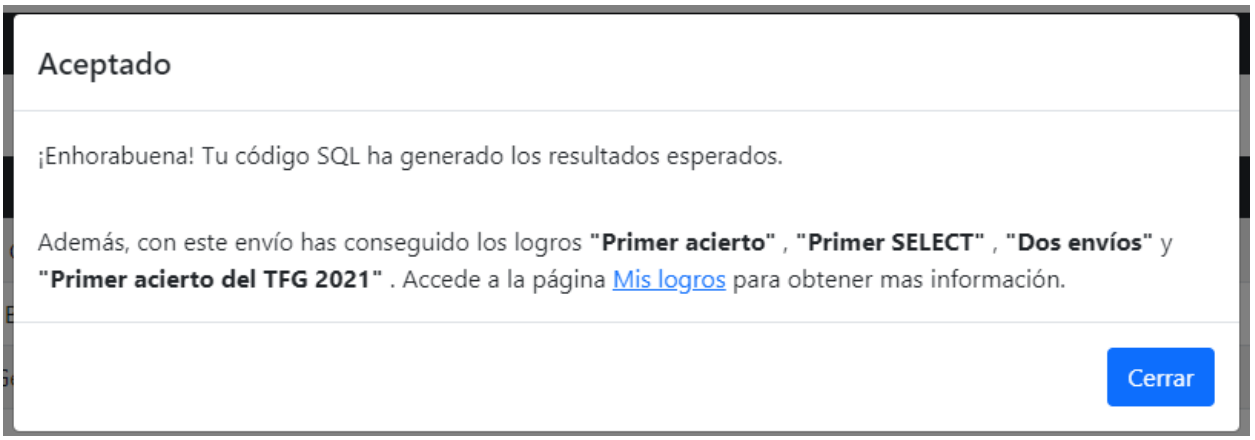
Las pruebas realizadas para esta tarea se han centrado en la petición de pistas, comprobando que el botón de solicitud de pista proporcionaba una pista correcta al usuario. Además, sobre una lista de pistas, se ha comprobado que se devuelven al usuario las pistas solicitadas en el orden adecuado cuando el problema tiene varias pistas disponibles. En la lectura del fichero `markdown` comprimido en un `ZIP` de las pistas disponibles para un problema, comprobando los mensajes que devuelven las excepciones que podría lanzar la función de lectura de pistas. Y por último, la comprobación de las tablas con el contenido correspondiente de las pistas usadas por un alumno.
Durante la ejecución de los test se realizan un total de 84 comprobaciones.

7.18 Informar al usuario de los logros conseguidos

Enlaces Pull Request: https://github.com/emartinm/lsql/pull/88
Enlace Issue: https://github.com/emartinm/lsql/issues/65

7.18.1 Descripción y motivación

Esta tarea consiste en mostrar al usuario un mensaje de información sobre la obtención de un logro en el momento realizar un envío, como se puede ver en la imagen:



A raíz de la creación de logros (apartado 7.13) era importante implementar un aviso para que así el usuario sepa el momento exacto en el que ha conseguido desbloquear

un logro tras haber realizado la misión correspondiente resolviendo un número específico de problemas o siendo uno de los primeros alumnos en resolver problemas. El objetivo de esta tarea es motivar a los alumnos al ver que tras resolver ejercicios y esforzarse en la asignatura, consiguen logros.

7.18.2 Desarrollo

El desarrollo de esta tarea consiste en añadir un mensaje informativo en el momento en el que realiza un envío correcto y ese envío desbloquea uno o varios logros. Este mensaje aparece en la ventana emergente que aparece al enviar cualquier envío e informa sobre el estado de ese envío, si es correcto o tiene algún error.

En la implementación hubo que modificar varias funciones, ya existentes, del sistema de logros y de la subida de soluciones de un problema. Se modificó también el fichero Javascript y se creó una plantilla para el mensaje de aviso. La principal dificultad de esta tarea fue cómo recorrer la lista de logros para poder mostrarlos en una frase separado por comas y no como si fuera una lista de elementos por puntos.

Durante el transcurso de la tarea se realizaron un total de 57 intercambios de mensajes, repartidos entre: 5 en el *Issue* 65 y 52 en el *Pull Request* 88.

7.18.3 Test Unitarios

Las pruebas realizadas para esta tarea se han centrado en el mensaje de alerta que se le muestra al usuario al conseguir un logro o la ausencia de este mensaje en caso de no haber desbloqueado ningún logro con el envío realizado.

Durante la ejecución de los test se han realizado un total de 5 comprobaciones.

8. Contribuciones

En esta sección se describen las contribuciones individuales de cada miembro del equipo al proyecto.

8.1 Iker Burgoa Muñoz

Desde que se empezó con el proyecto en septiembre, al haber una gran cantidad de tareas de primera instancia se hizo un reparto de tareas para que cada integrante del equipo se pusiera manos a la obra.

Mi primera tarea seleccionada fue la descrita en el apartado 7.1: añadir una opción en cada problema para exportar un archivo con extensión `SQL` que incluya las sentencias de creación de tablas e inserciones en dichas tablas para reproducir la base de datos del problema. Al principio, la curva de aprendizaje fue elevada, debido a que no había trabajado mucho con las herramientas elegidas y no había usado nunca Django. Una vez entendida gran parte de la documentación y comprendido el funcionamiento de la aplicación, la tarea se pudo resolver de manera satisfactoria.

Al ser el primer integrante en terminar la tarea, pude elegir la siguiente tarea entre todas las disponibles. La tarea del apartado 7.2 me pareció una elección interesante, ya que, como alumno de la asignatura, me interesaba que los mensajes que devolviera la aplicación fueran claros y se pudieran encontrar rápidamente fallos. La curva de aprendizaje fue menor, al tener conocimiento en la aplicación. Lo costoso fue encontrar y averiguar en qué archivos se podía encontrar la retroalimentación de los ejercicios. Una vez encontrado, no supuso gran dificultad llevar la tarea a su fin junto a sus pruebas unitarias.

La siguiente actividad requirió una labor de investigación (apartado 7.3), ya que al desconocer el gran potencial que aportaba Django con la creación de grupos y usuarios, era necesario conocer sus recursos, pues a la hora de organizarse profesores y alumnos, vendría de gran ayuda y liberaría trabajo, ya que esas funcionalidades estarían implementadas.

Las próximas tareas, los apartados 7.4 y 7.5, fueron las relativas a la creación de la propia vista de clasificación, en la que el alumno pudiera echar un vistazo rápido tanto de sus resultados como de los de sus compañeros en esa colección. Esto le permitía al profesor acceder a información de la evolución de la asignatura y saber el porcentaje de trabajo hecho por su alumnado. Este punto fue un proceso de aprendizaje superior al resto, pues mezclaba también Javascript y control de acceso a las páginas para que no hubiera fugas de información.

Una vez terminada la tarea anterior, se eligió como siguiente tarea un suplemento que permitiría que cada alumno accediera a los envíos filtrando por ejercicios concretos (apartado 7.5). Esto sería beneficioso también para el profesor, al permitirle un mayor control de los envíos concretos de los ejercicios y poder comprobar envíos más rápidamente.

La última tarea seleccionada fue exclusiva para los profesores y consistió en un filtro en la propia vista de clasificación donde poder filtrar la tabla de resultados por fechas (apartado 7.9). Era un elemento muy útil ya que se podía con ello analizar los ejercicios hechos en un momento determinado del curso y así ver la evolución de los alumnos por etapas.

En esta última tarea el tiempo de desarrollo fue sensiblemente menor, ya que el conocimiento previo de Javascript por las tareas anteriores sirvió para finalizar la tarea más rápidamente. Se realizó también un control de acceso ya que se añadió a la URL dos nuevos parámetros para las fechas y sus controles. El único punto que se complicó fue la creación de un formulario de Django para las fechas, ya que desconocía su funcionamiento, pero una vez controlado el modelo facilitó el control de las mismas y desde un mismo sitio se controlaba los mensajes de error que se pudieran producir.

8.2 Tamara Huertas Smolis

Una vez escogido este proyecto como TFG, tuvimos una primera reunión para una primera toma de contacto con el ámbito del proyecto y ver qué podíamos aportar nosotros al proyecto. Después de unas semanas hicimos varias propuestas de nuevas funcionalidades para incorporarlas al juez. Mis propuestas fueron hacer un botón para

descargar el código de envío de los ejercicios y un sistema de pistas que pudiera proporcionar alguna ayuda a los alumnos.

El sistema de software del que parte este TFG está desarrollado utilizando múltiples herramientas sofisticadas que en mi caso plantearon diversos problemas de instalación y configuración. Al instalar todo el entorno tuve complicaciones con la instalación de *Instant Client* de Oracle 11g y con las variables de entorno. Una vez solucionados estos problemas tuve que dedicarle tiempo a entender cómo funcionaba Django y ver algún tutorial de Python ya que era un *framework* y un lenguaje de programación que nunca había usado.

Mi primera tarea a desarrollar fue una que propuse yo en las primeras reuniones del equipo del proyecto: descargar el código de los envíos que realiza un alumno para cualquier problema (apartado 7.7). Esta tarea me parecía interesante porque por experiencia en el uso de otros jueces, al enviar una solución de un ejercicio, ese código no lo puedes ver en ninguna parte. Así también facilita a los alumnos el no tener que estar guardando esos ejercicios al cambiar de ordenador desde la facultad a su ordenador personal.

Al ser la primera tarea fue más complicada ya que tenía que aprender primero cómo funciona todo el proyecto y cómo está organizado para empezar a introducir tu código.
Aunque la curva de aprendizaje de todas las herramientas utilizadas en el proyecto es larga, en cuanto se empiezan a manejar la tarea no fue tan complicada. Lo que más difícil me resultó fueron los test unitarios de prueba porque nunca había trabajado con ellos. Pero después de investigar y ver ejemplos, pude completar la tarea.

Mi segunda tarea fue descargar la tabla de clasificación que realizó Iker en una hoja de cálculo (apartado 7.12). Esta tarea la escogí porque me parecía de gran ayuda para los profesores que pudieran descargarse los resultados finales de un curso y así poder ir haciendo un balance de cada curso y ver cómo van evolucionando los alumnos. La complicación de esta tarea fue encontrar una librería Python para poder descargar información en formato Excel. Probé varias librerías pero me costó encontrar una con la que poder tanto escribir en un fichero temporal como leer de él la información.

Mi siguiente tarea fue la de más peso, el sistema de pistas (apartado 7.17). Esta tarea fue la segunda que propuse en las primeras reuniones de coordinación del proyecto. Es la tarea que más ganas tenía de hacer ya que en ningún juez usado a lo largo de la carrera me he encontrado con algo similar. Era una tarea que me parecía muy interesante como alumna. Me parece una opción bastante buena para los alumnos que están un poco perdidos en la asignatura y que poco a poco quieren ponerse al día pero necesitan una pequeña ayuda. También para aquellos que encuentren dificultades adicionales para resolver algún ejercicio. Y todo esto sin necesidad de tener que esperar días hasta tener clase otra vez o tener que comunicarse con el profesor a través del correo, que muchas veces se acaba complicando por no saber cómo expresar tus dudas. Además, esta tarea contaría con dos funcionalidades adicionales: una vista para el alumno que muestra la recopilación de todas las pistas usadas para los diferentes problemas del juez, una opción para el profesor que consiste en permitir cargar las pistas directamente desde un fichero almacenado en la carpeta comprimida de cada problema de manera que una vez se lee la pista la añade al sistema.

Esta tarea fue bastante complicada ya que tenía que usar Javascript y no estaba nada familiarizada con ese lenguaje. Tuve que aprender múltiples aspectos de programación en Javascript, como por ejemplo hacer llamadas asíncronas a funciones, manejar diccionarios y JSON en Python, o crear plantillas HTML para añadir contenido en una página. Por otro lado, también tuve que investigar sobre la agregación de atributos dinámicos a un modelo y el uso de señales para descubrir cómo poder acceder a un atributo dinámico de un modelo.

Mi última tarea realizada fue la de informar a los usuarios al haber desbloqueado un logro (apartado 7.18) cuando superan ciertas misiones realizando envíos de problemas en el sistema de logros (apartado 7.13). Esta funcionalidad me parecía bastante interesante para los alumnos ya que sirve de motivación para seguir resolviendo ejercicios y conseguir más logros.

Esta tarea no fue muy complicada ya que tenía más experiencia al usar javascript, crear plantillas HTML y la renderización de estas plantillas, tras haber realizado las

tareas anteriores. El principal inconveniente fue como listar de manera correcta los logros conseguidos por el alumno.

8.3 Daniel Ibáñez Padial

Al inicio del proyecto se organizó una reunión de coordinación para hacer una lluvia de ideas y añadir nuevas funcionalidades, mejoras e implementaciones que se pudieran hacer sobre la versión inicial del sistema. Algunas de mis aportaciones para tareas a incluir fueron el sistema de logros, un chat para hablar con los profesores o un campo para agregar un archivo local y enviarlo como solución. De estas tareas la única que se desestimó para este Trabajo de Final de Grado fue la del chat para hablar con los profesores.

Tras el reparto inicial de tareas quedé contento con mi tarea asignada para empezar el proyecto, “Campo para enviar un archivo como respuesta” (apartado 7.8), ya que es una de las tareas que se me ocurrió a mí y la veía como un complemento muy positivo para este corrector automático, pues los alumnos iban a poder trabajar en un editor de textos en su ordenador local y después subirlo a la plataforma sin miedo a copiar elementos invisibles que a veces aparecen en los editores de texto.

Para realizar esta tarea tuve que refrescar mis conocimientos con Javascript y aprender nuevos conceptos, ya que debía conocer el funcionamiento de Ace Editor para que al subir el archivo local se mostrara el texto SQL para que el alumno pudiera editarlo antes de enviarlo.

Tras finalizar esta tarea y al ir bien de tiempo se nos asignó otra tarea etiquetada como dificultad baja, para ir familiarizándonos con la herramienta. Era la primera vez que trabajaba con Python y con el *framework* de Django, así que el primer mes fue completamente de aprendizaje a través de tutoriales y documentación. Una vez que aprendí los conceptos básicos y por dónde empezar me puse con mi segunda tarea, “Mostrar tablas modificadas” (apartado 7.11), mejorar las tablas mostradas en el apartado de resultado esperado, dentro de la vista de los problemas. Se añadió una etiqueta al lado de cada tabla modificada, eliminada o agregada. Esto podría ayudar mucho a los alumnos al realizar los ejercicios de Trigger, Procedimientos y DML, ya

que al ver el resultado esperado con más detalle sabrán cómo arrancar con el código. Durante la realización de la tarea tuve que volver a mirar alguna documentación de HTML para poder crear las nuevas vistas de estas tablas.

Después de haber integrado esta tarea con la rama principal del TFG, empecé con mi tarea del sistema de logros (apartado 7.13), en mi opinión una de las tareas más interesantes, ya que creo que era una de las más positivas para los alumnos. Con este sistema de logros se puede conseguir que los alumnos sientan una motivación especial para realizar los ejercicios, poniendo los logros como una meta para alcanzar mientras aprenden. Tuve que leer mucha más documentación sobre Django y sus modelos, además de introducir por primera vez en este juez las señales, algo que todavía no estaba integrado pero que es una herramienta muy potente de este *framework*. Además también tuve que cambiar los estilos del CSS para poder modificar los colores del trofeo que debía aparecer en el ranking.

Al terminar esta tarea pude elegir mi última tarea, la cual decidí que fuera el nuevo tipo de problemas, “Ejercicios de discriminación de consultas” (apartado 7.15). Como se ha podido observar todas mis contribuciones han ido dirigidas a los alumnos, ya que creo que esta herramienta debe ir enfocada a que los alumnos continúen con ganas de seguir aprendiendo y con una motivación extra. Esta última tarea no iba a ser diferente, creo que agregar un nuevo tipo de problemas es algo muy positivo: además de aprender otra parte de SQL, hace que no sean tan monótonos los ejercicios propuestos. Es un tipo de ejercicio que no he visto en ningún otro juez y que creo que puede mostrar a los alumnos que no todas las respuestas, aunque sean correctas para una base de datos particular, son válidas.

Conseguí hacer esta tarea en poco tiempo así que solicité a los tutores la posibilidad de hacer una nueva tarea, agregando nuevos logros al sistema creado por mí anteriormente (apartado 7.13). Esto lo hice para seguir ampliando el abanico de posibilidades para los profesores de hacer nuevos logros para que los alumnos puedan seguir resolviendo ejercicios en busca de nuevos logros que puedan conseguir.

8.4 Iván Ruiz Quintana

El comienzo del proyecto fue una tarea un tanto frustrante, ya que antes de poder empezar con las tareas que nos fueron asignadas había que explorar y entender todo el funcionamiento de la aplicación para poder comenzar con el trabajo así como aprender y buscar documentación sobre Django, ya que nunca antes había programado usando este *framework*. El proyecto contaba con muchos archivos y líneas de código que teníamos que entender, pero con las instrucciones, las explicaciones que nos proporcionaron, nuestro esfuerzo y tiempo, nos fuimos acostumbrando a ir añadiendo y modificando funcionalidades.

Mi primera tarea consistió en añadir un nuevo botón en la interfaz para mostrar/ocultar tablas en los ejercicios (apartado 7.6). Era una funcionalidad que afectaba exclusivamente a la interfaz de usuario del sistema, relacionada únicamente con HTML y CSS por lo que no me resultó muy complicada ya que tenía recientes estos conocimientos gracias a las asignaturas cursadas este año. Me recomendaron usar *Bootstrap* para añadir esta función de manera más sencilla y limpia por lo que tuve que investigar sobre ello ya que no lo había utilizado anteriormente. Tras completar la tarea llegó el momento de subir los cambios a través de GitHub, herramienta que no había usado antes y por tanto también me llevó unos días aprender su funcionamiento. Una vez adquiridos los conocimientos para usar GitHub, no tuve que preocuparme más en las siguientes tareas.

La siguiente tarea fue más complicada que la anterior, consistía en añadir un podio con los primeros tres usuarios en completar cada problema. En esta tarea tuve que profundizar en la estructura del sistema más allá de la interfaz, a diferencia de la tarea anterior. Me llevó unos días entender bien cómo funcionaba todo el sistema de los problemas y de qué manera se guardaban todos los datos para así poder seleccionar los usuarios que se debían mostrar para seguir con la realización de mi tarea. Una vez comprendido el funcionamiento, pude terminar la tarea y comenzar con la creación de los tests, lo cual era nuevo para mí ya que en la tarea anterior no fue necesario crearlos. Tras investigar, estudiar la documentación sobre la creación de tests, y revisar los ya existentes, conseguí entender cómo se llevaban a cabo y los pude hacer de

manera bastante intuitiva y así no tener más problemas a la hora de crearlos en las siguientes tareas.

Tras realizar esta tarea se me encomendó la siguiente, añadir la posibilidad de usar más de una base de datos inicial en la corrección de los problemas. Esta tarea fue puramente interna ya que no habría que añadir nuevas funciones a la interfaz como tal, solo habría que modificar el comportamiento interno del juez. A primera vista me pareció una tarea complicada pero, tras aprender el funcionamiento de cómo el juez realiza las comprobaciones y evalúa las soluciones enviadas por los alumnos, pude llevar a cabo la nueva implementación de tal manera que quedé bastante satisfecho con su funcionamiento. En esta tarea, al ser una nueva funcionalidad bastante delicada, tuve que centrarme más en los tests que en las anteriores, ya que manejaba aspectos relacionados con la corrección de problemas, la principal característica del proyecto, y no podía tener ningún fallo.

Como último desarrollo, elegí incorporar un soporte multidioma para facilitar su uso a estudiantes extranjeros o su futuro uso en diferentes idiomas. Me pareció una idea bastante interesante, además de tener experiencia previa por mi parte en el uso de múltiples idiomas en una aplicación. Esta fue la tarea más extensa de las que he realizado en el proyecto y al mismo tiempo la que más me gustó. Al tener la experiencia necesaria para realizar la tarea gracias a todo el trabajo anterior a esta, pude emplear la mayor parte del tiempo en incluir las modificaciones y crear los tests y menos tiempo en informarme y buscar documentación. Aun así me llevó tiempo aprender cómo funcionaba el soporte de múltiples idiomas en Django e incorporarlo al proyecto, pero, una vez hecho esto, el resto de la tarea llevó una trayectoria menos compleja.

9. Conclusiones y trabajo futuro

A vista de los resultados obtenidos y al trabajo desarrollado, se pueden sacar varias conclusiones.

La primera conclusión es la importancia de una herramienta potente en asignaturas relacionadas con Bases de Datos. Al estudiante le aporta una forma distinta de estudio y sobre todo una motivación extra, ya que las nuevas funcionalidades que fomentan la participación del alumno son especialmente interesantes. La obtención de logros, podio y la clasificación de los alumnos persigue este objetivo, y el sistema de pistas puede ayudar a mantener la participación de los alumnos en los ejercicios más complejos.

Por parte del profesor, le otorga otro punto de vista de impartir la asignatura y tener toda la información sobre los ejercicios realizados, los envíos realizados y el número de ejercicios recogida en una herramienta, lo cual puede influir en la forma de orientar la asignatura durante el transcurso del curso.

El uso de la metodología *Open Source* y la herramienta de GitHub puede ser una buena elección para el desarrollo de un proyecto ya que otorga muchas facilidades. Aporta una visión de un método de desarrollo nuevo, que podría utilizarse para futuros desarrollos de proyectos o trabajos de final de grado.

En el apartado 1.2 se plantearon una serie de objetivos concretos. Se puede afirmar que con este trabajo, se han conseguido alcanzar estos objetivos con las tareas planteadas durante el desarrollo del trabajo. En la siguiente lista se muestran las tareas que han permitido satisfacer cada uno de los objetivos planteados inicialmente:

- Mejoras de usabilidad de la aplicación.
    a) Botón para descargar los script de las tablas (Apartado 7.1).
    b) Mejorar la respuesta generada ante una solución incorrecta en el esquema generado (Apartado 7.2).
    c) Enlaces de los envíos de la tabla de clasificación (Apartado 7.5).
    d) Colapsar las tablas (Apartado 7.6).
    e) Descargar los envíos realizados (Apartado 7.7).

f) Campo para enviar un archivo como respuesta (Apartado 7.8).
g) Mostrar las tablas modificadas (Apartado 7.11).
h) Descargar el ranking de la clase (Apartado 7.12).

- Funcionalidades y mejoras para aumentar la motivación del alumno y la ludificación.
    a) Vista de los resultados de la clase (Apartado 7.4).
    b) Filtro de fechas en la vista de resultados (Apartado 7.9).
    c) Podio (Apartado 7.10).
    d) Logros (Apartado 7.13).
    e) Tarea sobre pistas (Apartado 7.17).
    f) Informar al usuario de los logros obtenidos (Apartado 7.18)
- Soporte de múltiples idiomas.
    a) Soporte multidioma (Apartado 7.16).
- Habilitación de múltiples grupos de clase.
    a) Grupos de clase (Apartado 7.3).
- Nuevos tipos de problemas y mejoras sobre los problemas existentes.
    a) Bases de datos iniciales (Apartado 7.14).
    b) Ejercicios de discriminación de consultas (Apartado 7.15).

Durante el desarrollo del trabajo han aparecido muchas otras nuevas funcionalidades y mejoras que se pueden implementar en este sistema, así como propuestas planteadas por otros profesores de la facultad, pero que no es posible realizar en el ámbito de este trabajo de fin de grado. Learn SQL es una herramienta que se ofrecerá a distintas universidades y será usada por por los directores de este trabajo en varios grupos de la asignatura Bases de Datos de la Facultad de Informática de la UCM en el curso académico 2021-2022 y ofrecerán, tanto alumnos como profesores, nuevas ideas que pueden convertirse en mejoras futuras como las nombradas a continuación:

1) Añadir a la herramienta un modo examen, ofreciendo un horario reducido, sin retroalimentación de los ejercicios y sin resultados esperados.
2) Crear sistema de mensajería interno en la herramienta donde el alumno pueda comunicarse directamente con el profesor.

3) Añadir en los problemas, sugerencias de otros problemas similares para poder seguir practicando con los mismos conceptos, incrementando su dificultad, o bien continuando con las siguientes nociones en el temario de la asignatura.
4) Que se puedan soportar otros sistemas de gestión de bases de datos, como por ejemplo MySQL, MariaDB…
5) Asociar a cada problema una puntuación que será usada en caso de resolver el problema, con el objetivo de que esos puntos sean utilizados para ordenar la clasificación de alumnos del grupo.
6) Realizar una medida de la complejidad de la consulta enviada por el alumno. Esta nueva funcionalidad puede ser muy compleja, pero se puede usar una métrica sencilla, como por ejemplo no utilizar tablas innecesarias de acuerdo con la solución oficial, o comprobar que se utiliza una técnica concreta (subconsultas, reuniones externas, etc.).
7) Utilizar analizadores de SQL más potentes, como los que usa el sistema DES [20] para mejorar los mensajes de error.

10. Conclusions and future work

In view of the results obtained and the work carried out during this project, several conclusions can be drawn.

The first conclusion is the importance of a powerful exercise correcting tool in subjects related to databases. It provides students a different way of learning in these subjects and, above all, it gives extra motivation, as the new functionalities that encourage student participation are particularly interesting. The achievements, podium and ranking of students pursue this goal, and the hint system can help to keep students engaged in the more complex exercises.

On the teacher's side, it gives them another approach to teaching this type of subjects, and having in a single tool all the information available about exercises carried out, submissions made and the number of exercises completed, can influence a great deal the way the subject is conducted during the course.

The use of the open source methodology and the GitHub tool can be a good choice for the development of a project as it provides many facilities. It provides an insight into a new development method, which could be used for future project development or final degree projects.

In section 1.2, a series of specific objectives were set out. It can be affirmed that those objectives have been met with the tasks set out during the development of this project. The following list shows the tasks that have allowed each of the initially proposed objectives to be met:

- Improvements to the usability of the application.
    a) Button to download the script of the tables (Section 7.1).
    b) Improve the generated response to an incorrect solution in the generated scheme (Section 7.2).
    c) Links to the submissions in the league table (Section 7.5).
    d) Collapse the tables (Section 7.6).
    e) Downloading the submissions made (Section 7.7).

f) Field for sending a file as a reply (Section 7.8).
g) Display the modified tables (Section 7.11).
h) Download the class ranking (Section 7.12).

- Functionalities and improvements to increase learner motivation and gamification.
  a) Class results view (Section 7.4).
  b) Date filtering in the results view (Section 7.9).
  c) Podium (Section 7.10).
  d) Achievements (Section 7.13).
  e) Hints (Section 7.17).
  f) Informing the user of achievements (Section 7.18).
- Multiple language support.
  a) Multiple language support (Section 7.16).
- Enabling multiple class groups.
  a) Class groups (Section 7.3).
- New problem types and improvements to existing problems.
  a) Initial databases (Section 7.14).
  b) Query discrimination exercises (Section 7.15).

During the development of the work, many other new functionalities and improvements have appeared that can be implemented in this system, as well as proposals put forward by other professors in the faculty, but that are unfeasible to implement within the scope of this final degree project. Learn SQL is a tool that will be offered to different universities and will be used by the directors of this work in several groups of the subject Databases of the Faculty of Computer Science of UCM in the academic year 2021-2022 and this will provide, from both students and teachers, new ideas that can become future improvements such as those mentioned below:

1) Add an exam mode to the tool, offering a reduced timetable, no feedback on the exercises and no expected results.
2) Create an internal messaging system in the tool where the student can communicate directly with the teacher.

3) Add in the problems, suggestions of other similar problems to be able to continue practising with the same concepts, increasing their difficulty, or continuing with the following notions in the syllabus of the subject.
4) Add support for other database management systems, such as MySQL, MariaDB, etc.
5) Associate to each problem a score that will be used in case of solving the problem, with the aim that these points will be used to sort the ranking of students in the group.
6) Perform a measure of the complexity of the query sent by the student. This new functionality can be very complex, but simple metrics can be used, such as “not using unnecessary tables according to the official solution”, or “checking that a particular technique is used” (subqueries, external meetings, etc.).
7) Use more powerful SQL parsers, such as those used by the DES system [20] to improve error messages.

Bibliografía

[1] Django Software Foundation. *Django*. https://www.djangoproject.com/

[2] Oracle. *Oracle Database*. https://www.oracle.com/es/database/

[3] PostgreSQL. https://www.postgresql.org/

[4] Python Software Foundation. *Pylint.* https://pypi.org/project/pylint/

[5] Codecov. https://about.codecov.io/

[6] JetBrains. *PyCharm*. https://www.jetbrains.com/es-es/pycharm/

[7] Notepad++. https://notepad-plus-plus.org/

[8] Sublime HQ Pty. *Sublime Text*. https://www.sublimetext.com/

[9] S. Chacon, B. Straub. *Pro Git*. https://git-scm.com/book/es/v2/Ramificaciones-en-Git-%C2%BFQu%C3%A9-es-una-rama%3F

[10] Balsamiq Studios, *LLC. Balsamiq*. https://balsamiq.com/

[11] Juanguis. *Frame Box, herramienta para crear mockups online*. https://www.puntogeek.com/2010/12/09/frame-box-herramienta-para-crear-mockups-online/

[12] Microsoft. *Paint*. https://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/mspaint_overview.mspx?mfr=true

[13] DOMjudge developers and contributors, (2004-2021), *DOMjudge* https://www.domjudge.org/docs/manual/7.3/index.html

[14] L. Llana, E. Martin Martin, C. Pareja Flores, J. Velazquez Iturbide, (2014), *FLOP: A User-Friendly System for Automated Program Assessment*. http://www.jucs.org/doi?doi=10.3217/jucs-020-09-1304

[15] Refsnes Data, (1999-2021), *w3School.* https://www.w3schools.com/sql/

[16] Oracle Corporation, (2021), *Oracle Live SQL.* https://livesql.oracle.com

[17] A. Cumming, (2020), *SQL Zoo.* https://sqlzoo.net/

[18] J. Feasel, (2012), *SQL Fiddle.* http://sqlfiddle.com/ https://sqlzoo.net/

[19] SQLBolt, (2019), *SQL Bolt* https://sqlbolt.com/

[20] F. Saenz-Perez, DISIA, FADoSS, UCM, (2004-2021), *DES (Datalog Educational System) V6.6*. https://www.fdi.ucm.es/profesor/fernan/des/

[21] Django Software Foundation. *Django Documentation*. https://docs.djangoproject.com/en/3.2/

[22] IANA Protocolos, *IANA*. https://www.iana.org/assignments/media-types/media-types.xhtml

[23] Mozilla. *Grupos y Usuarios Django*. https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Authentication

[24] Django Software Foundation. *Django Request and response objects*. https://docs.djangoproject.com/en/3.2/ref/request-response/

[25] Bootstrap. *Bootstrap Collapse*. https://getbootstrap.com/docs/4.0/components/collapse/

[26] ACE. *Ace Editor*. https://ace.c9.io/

[27] Django Software Foundation. *Django Forms*. https://docs.djangoproject.com/en/3.2/topics/forms/

[28] Django Software Foundation. *Django QuerySet*. https://docs.djangoproject.com/en/3.1/ref/models/querysets/

[29] Font Awesome. https://fontawesome.com/v6.0/icons

[30] E. Gazoni y C. Clark, (2018), *openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files.* https://openpyxl.readthedocs.io/en/default/

[31] L. Richardson, (2020), *Beautiful Soup 4.9.0 Documentation.* https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[32] Python Software Foundation, (2021), *tempfile — Generate temporary files and directories*. https://docs.python.org/3/library/tempfile.html#tempfile.mkstemp

[33] Django Software Foundation. *Django Models*. https://docs.djangoproject.com/en/3.2/topics/db/models/

[34] Django Software Foundation. *Django Signals*. https://docs.djangoproject.com/en/3.2/topics/signals/

[35] Django Software Foundation. *Django Translation*. https://docs.djangoproject.com/en/3.2/topics/i18n/translation/

[36] Lipis. *Flag-icon-css*. https://github.com/lipis/flag-icon-css

[37] Django Software Foundation. (2005-2021) *Django Javascript*. https://docs.djangoproject.com/en/dev/internals/contributing/writing-code/javascript/

Otras fuentes consultadas

A lo largo del proyecto se han utilizado otras páginas web o aplicaciones para realizar el desarrollo del trabajo:

1 - StackOverflow. https://es.stackoverflow.com/

2 - Mozilla Developer. https://developer.mozilla.org/es/

3 - YouTube. https://www.youtube.com/

4 - Delf Stack. https://www.delftstack.com/es/

5 - Código Pitón. https://www.codigopiton.com/

6 - El Libro de Python. https://ellibrodepython.com/